FINAL EXAM: DATABASES ("BASES DE DATOS") – 9/06/06 – SCHEMA

Consider the following relational schema, which will be referred to as WORKING SCHEMA, which maintains information on the order and invoices of a retail company:

CUSTOMER (cust_code: dom_cust_code, **name**: dom_name, **NIF**: dom_NIF, **telephone**: dom_tel, **town**: dom_town**)**
PK: {cust_code}
NNV: {name, NIF, telephone}
UNIQUE: {NIF}

ARTICLE (art_code: dom_art_code, **description**: dom_desc, **price**: dom_price**)**
PK: {art_code}
NNV: {price}

ORDER (ord_code: dom_ord_code, **date**: dom_date, **cust_code**: dom_cust_code, **amount**: dom_amount**)**
PK: {ord_code}
FK: {cust_code} → CUSTOMER
Restrictive delete. On update cascade. Deferrable initially immediate
VNN {cust_code}

ORDER_LINE (ord_line: dom_ord_line**, ord_code**: dom_ord_code**, quantity**: dom_cant**, art_code**: dom_art_code**)**
PK: {ord_line, ord_code}
NNV: {quantity, art_code}
FK: {art_code} → ARTICLE
Restrictive delete. On update cascade. Deferrable initially immediate
FK: {ord_code} → ORDER
Restrictive delete. On update cascade. Deferrable initially immediate

INVOICE (inv_code: dom_inv_code, **date**: dom_date, **cust_code**: dom_cust_code, **amount**: dom_amount**)**
PK: {inv_code}
NNV: {cust_code, date}
FK: {cust_code} → CUSTOMER
Restrictive delete. On update cascade. Deferrable initially immediate

INVOICE_LINE (inv_line: dom_inv_line, **inv_code**: dom_inv_code, **ord_line**: dom_ord_line, **ord_code**: dom_ord_code**)**
PK: {inv_line, inv_code}
NNV: {ord_line, ord_code}
FK: {inv_code} → INVOICE
Restrictive delete. On update cascade. Deferrable initially immediate
FK: {ord_line, ord_code} → ORDER_LINE
Complete referential integrity
Restrictive delete. Restrictive update. Not deferrable
UNIQUE: {ord_line, ord_code}

where the attributes and tables have the following meaning

CUSTOMER: description of the customers of the company

cust_code, *name*, *NIF, telephone* and *town* represent the code, name, NIF, telephone and town of the customer.

ARTICLE: description of the articles sold by the company

art_code: code of the article,
description: description,
price: price.

ORDER: description of the orders which are requested to the company

ord_code: code of the order,
date: date in which the order is received
cust_code: code of the customer who makes the order,
amount: amount (in euros) of the order (sum of *price*quantity* of all the lines in the order)

ORDER_LINE: description of each line in each order

ord_line: code of the line in the order
ord_code: code of the order where the line is
art_code: code of the article which is requested in the line
quantity: number of units (in kg., m., etc.) in which the article is measured

INVOICE: description of the invoices which are performed by the company

inv_code: code of the invoice
date: date of the invoice
amount: amount (in euros) of the invoice (sum of *price*quantity* of all the lines in the invoice)
cust_code: code of the customer to whom the invoice is made

INVOICE_LINE: description of each line in each invoice

inv_line: code of the line in the invoice,
inv_code: code of the invoice where the line is,
ord_line, ord_code: indicate which line of an order is paid by the line of the invoice

And consider the following extension of the previous schema. We will refer to this extension as database (DB):

CUSTOMER					ARTICLE		
cust_code	name	NIF	telephone	town	art_code	description	price
c1	Luisa	111	111111	Valencia	a1	patatas	1
c2	Juan	222	222222	Torrente	a2	manzanas	1.5
c3	Pepe	333	333333	Valencia	a3	pollo	3

ORDER				INVOICE			
ord_code	date	cust_code	amount	inv_code	date	cust_code	amount
p1	29/05/2006	c1	5	f1	31/05/2006	c1	9.5
p2	30/05/2006	c2	8.5	f2	1/06/2006	c2	8.5
p3	30/05/2006	c1	4.5				

ORDER_LINE				INVOICE_LINE			
ord_line	ord_code	art_code	quantity	inv_line	inv_code	ord_line	ord_code
1	p1	a1	2	1	f1	1	p1
2	p1	a3	1	2	f1	2	p1
1	p2	a1	1	3	f1	1	p3
2	p2	a2	1	4	f1	2	p3
3	p2	a3	2	1	f2	1	p2
1	p3	a1	3	2	f2	2	p2
2	p3	a2	1	3	f2	3	p2

Final exam: "databases" – 9/06/06 – QUESTIONNAIRE TYPE A

This questionnaire has 14 questions; for each one we propose four possible answers. Only one of them is correct. The answer must be included in the answer sheet which has been handed with the exam. The maximum mark for the questionnaire is 3.5 points. The result is obtained through the formula: (Right – Wrong/3) × 0.25.

1. Which would the final state be of the tables *Order, Order_line* and *Invoice_line* after the execution of the instruction "UPDATE Order SET ord_code = p4 WHERE ord_code = p1" over the database DB?

a)

ORDER			
ord_code	**date**	**cust_code**	**amount**
p4	**29/05/2006**	**c1**	**5**
p2	**30/05/2006**	**c2**	**8.5**
p3	**30/05/2006**	**c1**	**4.5**

The rest would remain the same

b)

ORDER			
ord_code	**date**	**cust_code**	**amount**
p4	**29/05/2006**	**c1**	**5**
p2	**30/05/2006**	**c2**	**8.5**
p3	**30/05/2006**	**c1**	**4.5**

The rest would remain the same

ORDER_LINE			
ord_line	**ord_code**	**art_code**	**quantity**
1	**p4**	**a1**	**2**
2	**p4**	**a3**	**1**
1	**p2**	**a1**	**1**
2	**p2**	**a2**	**1**
3	**p2**	**a3**	**2**
1	**p3**	**a1**	**3**
2	**p3**	**a2**	**1**

c)

ORDER			
ord_code	**date**	**cust_code**	**amount**
p4	**29/05/2006**	**c1**	**5**
p2	**30/05/2006**	**c2**	**8.5**
p3	**30/05/2006**	**c1**	**4.5**

ORDER_LINE				INVOICE_LINE			
ord_line	**ord_code**	**art_code**	**quantity**	**inv_line**	**inv_code**	**ord_line**	**ord_code**
1	**p4**	**a1**	**2**	**1**	**f1**	**1**	**p4**
2	**p4**	**a3**	**1**	**2**	**f1**	**2**	**p4**
1	**p2**	**a1**	**1**	**3**	**f1**	**1**	**p3**
2	**p2**	**a2**	**1**	**4**	**f1**	**2**	**p3**
3	**p2**	**a3**	**2**	**1**	**f2**	**1**	**p2**
1	**p3**	**a1**	**3**	**2**	**f2**	**2**	**p2**
2	**p3**	**a2**	**1**	**3**	**f2**	**3**	**p2**

d) Nothing is changed because the instruction violates the integrity constraint which is defined on the table *Invoice_line* over the foreign key which refers to *Order_line*.

2. Given the following integrity constraint:

```
CREATE ASSERTION r1 CHECK
(NOT EXISTS (SELECT * FROM Invoice F
    WHERE EXISTS (SELECT * FROM Invoice_line L
      WHERE L.inv_code = F.inv_code AND
            EXISTS (SELECT * FROM Invoice_line L1
                WHERE L.inv_code=L1.inv_code
                AND L.ord_code<>L1.ord_code))));
```

Which consequences would it have if we had included it in the definition of the working schema?

a) Would allow us to have several lines in the same invoice with different orders.
b) Wouldn't allow us to have several lines in the same invoice with different orders
c) This constraint cannot be defined in standard SQL.
d) This constraint is redundant since it is already taken into account by the schema.

3. Given the working schema. Which of the following assertions is TRUE?

a) We can delete a customer if s/he has no orders.
b) We can always delete a customer.
c) We can delete a customer if s/he has no invoices.
d) We can delete a customer if s/he has neither orders nor invoices.

4. Which is the result of the following expression in Relational Algebra?:

$$CUSTOMER[cust\_code,name] - (CUSTOMER[cust\_code, name] \bowtie ORDER[cust\_code])$$

a) It is syntactically incorrect, so no result will be returned.
b) The name and the code of all the customers who have orders in the database.
c) The name and the code of the customers who haven't performed all the orders of the database.
d) The name and the code of the customers who haven't performed any order.

5. About the definition of the relation *Invoice_line* in the working schema, which of the following options is TRUE?

a) The type of referential integrity of the foreign key to *Order_line*, FK: {ord_line, ord_code}, could be removed without any consequence since these attributes have a not null value constraint.
b) The constraint NNV: {ord_line, ord_code} could be removed without any consequence since these attributes are a foreign key with complete referential integrity.
c) Since {ord_line, ord_code} have a not null value constraint and a uniqueness constraint, these attributes should be the primary key of the relation.
d) Given that {ord_code} is the primary key of Order, we should add FK: {ord_code} $\rightarrow$ Order so that the relation would be well defined.

6. Given the database DB and the transaction:

```
COMMIT;
INSERT INTO Customer VALUES ('c4', 'Pepe', '333', 55555, 'Castellón');
INSERT INTO Article VALUES ('a4', 'peras', 1.5);
COMMIT;
```

Which of the following options is TRUE if we execute this transaction in ORACLE?

a) The system would only insert a customer.
b) The system would only insert the article.
c) The system would insert the customer and the article.
d) The system would insert neither the customer nor the article.

7. Given the working schema, which of the following assertions is FALSE?

a) An invoice can correspond to a customer who hasn't performed any order.
b) An order can be associated with more than one invoice.
c) All the lines in an invoice are associated to the same customer.
d) There can be some lines in an order which are not associated to any invoice.

8. Assume the following general integrity constraint is defined in the working schema in standard SQL:

```
(CREATE ASSERTION R1 CHECK(
   NOT EXISTS(SELECT * FROM CUSTOMER C WHERE
                  NOT EXISTS (SELECT * FROM ORDER P
                                 WHERE P.cust_code=C.cust_code)))
   NOT DEFERRABLE);
```

Which of the following transactions would allow the insertion of a new customer?

a)
```
COMMIT;
SET CONSTRAINT ALL DEFERRED;
INSERT INTO CUSTOMER VALUES ('c11', 'Pepe', 2090, 96345, 'Valencia');
INSERT INTO ORDER VALUES ('p11', '1/1/2006', 'c11', 0);
COMMIT;
```
b)
```
SET CONSTRAINT ALL DEFERRED;
COMMIT;
INSERT INTO CUSTOMER VALUES ('c11', 'Pepe', 2090, 96345, 'Valencia');
INSERT INTO ORDER VALUES ('p11', '1/1/2006', 'c11', 0);
COMMIT;
```
c) The insertion of new customers is not possible if the general integrity constraint is included in the database schema.
d)
```
COMMIT;
SET CONSTRAINT ALL DEFERRED;
INSERT INTO ORDER VALUES ('p11', '1/1/2006', 'c11', 0);
INSERT INTO CUSTOMER VALUES ('c11', 'Pepe', 2090, 96345, 'Valencia');
COMMIT;
```

9. From the following instructions in SQL, which one CANNOT be used to define an external schema?

a) CREATE VIEW.
b) DROP TABLE.
c) GRANT.
d) CREATE TRIGGER

10. If the database which has been proposed in the working schema requires frequent joins between the relations *Order and Order_line*, which file organisation is more adequate to store these relations?

a) Each relation in a file with a primary index over the attributes which are defined as primary key.
b) Each relation in a sequential file which is ordered by the primary key.
c) Store the relations in a "cluster", using as the clustering key the attribute *ord_code*.
d) A hashed organisation based in the search attribute *ord_code*.

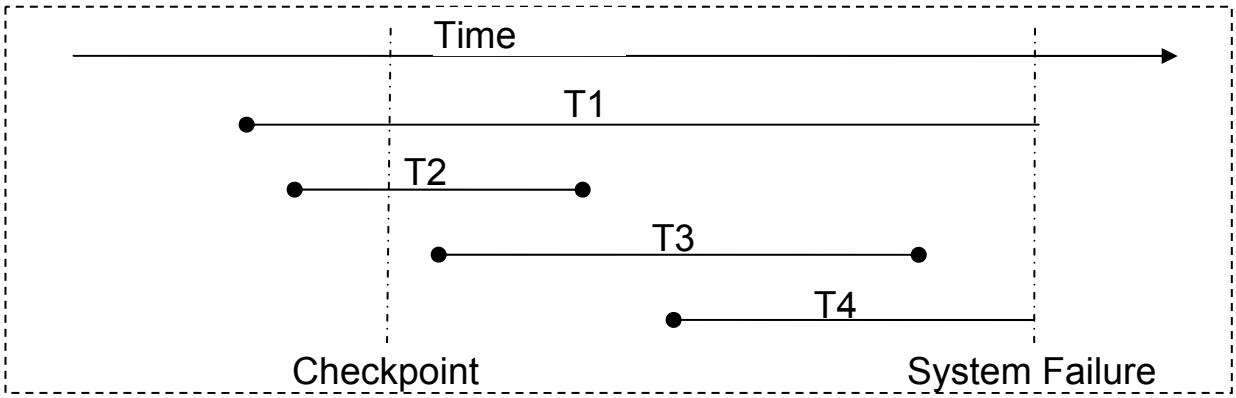
11. Given the working schema, what will the maximum and minimum cardinality be for the following query?: *Order_line* $\triangleright\triangleleft$ *Invoice_line*

a) The minimum and maximum cardinality match and it is Card(Order_line) $\times$ Card(Invoice_line).
b) The minimum and maximum cardinality match and it is Card(Invoice_line).
c) The minimum and maximum cardinality match and it is Card(Order_line).
d) The minimum cardinality is Card(Order_line) and the maximum cardinality is Card(Order_line) $\times$ Card(Invoice_line).

12. Physical independence is the level of independence which is established:

a) Between application programs which are written by users such that some users are not affected by other users.
b) Between the internal schema and the external schemas so that the latter are not affected by changes which refer to the logical data structures.
c) Between the logical schema and the internal schema so that the changes which are performed over the logical schema do not entail a modification of the organisations which have been chosen for the files which implement the database.
d) Between the logical schema and the internal schema so that the logical schema is not affected by changes in the internal schema which refer to different implementation details.

13. In the following temporal transaction diagram in a DBMS with immediate update, in which T2 and T3 have been confirmed, what should the DBMS do when the system is restarted after a system failure with loss of main memory?



a) Undo T4
b) Redo T1, T2 and T3, and undo T4.
c) Redo T2 and T3 and undo the rest.
d) Redo T3 and undo T1 and T4.

14. A computerised information system:
a) is a system which is exclusively composed by the database and the database management system which manipulates it.
b) always has an associated database which gives support to the system.
c) is an information system which is supported on one or more computers.
d) is a structured data collection which is useful to the organisation.

FINAL EXAM: DATABASES – 09/06/06 – Problems

Given the working schema presented before, solve the following exercises in standard SQL:

1) Obtain the *name* and the *telephone* number of the customers for which we do not know their town and whose name begins with 'A'. (0.5 points)

2) Obtain the *name* and *telephone* of the customers who haven't ordered anything in the year 2005. (0.5 points)

3) Obtain the attribute *ord_code* of those orders in June of 2005 which have no invoice and which are requested by customers located in Valencia. (0.5 points)

4) Obtain the codes of the orders, having at least one order line, such that in all their lines we have a *quantity* greater than 3. (1 point)

5) Obtain the attribute *ord_code* of those orders which have several lines in which the same article appears. (0.75 points)

6) Create a general constraint in SQL (ASSERTION) to ensure that every invoice has at least one invoice line. (1 point)

7) Obtain the *name* of the customer who, accumulating all his/her invoices, has been invoiced more money. Also obtain the *amount* of all his/her invoices. (1 point)

8) The attribute *amount* of the relation *Order* is a derived attribute which is obtained by summing up the result of multiplying the *price* of the article which appears in the line of the order by the requested *quantity*, for all the lines in the order, i.e.:

$$Amount_{\text{for the order i}} = \sum_{\text{for each line l in order i}} \left(\langle \text{price of the article of line l} \rangle \times \langle \text{quantity of line l} \rangle \right)$$

a. Apart from the deletion on *Order_line*, indicate five operations which affect the value of the derived attribute *amount* in the relation *Order*. (0.5 points)
b. Write a trigger to handle the deletion operation on *Order_line*. (0.75)

SOLUTIONS TO THE QUESTIONNAIRE:

1	d)
2	b)
3	d)
4	d)
5	a)
6	b)
7	c)
8	d)
9	b)
10	c)
11	b)
12	d)
13	c)
14	b) / c)

SOLUTIONS TO THE PROBLEMS:

1)

```
SELECT name, telephone
FROM Customer
WHERE town IS NULL AND name LIKE 'A%';
```

2)

```
SELECT name, telephone
FROM customer
WHERE cust_code NOT IN (SELECT cust_code
                        FROM order
                        WHERE date BETWEEN '01/01/2005' AND '31/12/2005');
```

3)

```
SELECT ord_code
FROM order p, customer c
WHERE p.cust_code=c.cust_code AND c.town = 'Valencia' AND
      p. date BETWEEN '1/06/2005' AND '30/06/2005' AND
      ord_code NOT IN (SELECT ord_code FROM invoice_line)
```

4)

```
SELECT ord_code
FROM Order p
WHERE NOT EXISTS (SELECT * FROM Order_line lp
                  WHERE lp.ord_code=p.ord_code AND lp.quantity<=3)
                        AND EXISTS (SELECT * FROM Order_line lp
                                    WHERE lp.ord_code=p.ord_code);
```

or (valid in standard SQL but not in Oracle):

```
SELECT ord_code
FROM Order p
WHERE FOR ALL (SELECT * FROM Order_line lp
                        WHERE lp.ord_code=p.ord_code) (lp.quantity>3)
                        AND EXISTS (SELECT * FROM Order_line lp
                        WHERE lp.ord_code=p.ord_code);
```

5)

```
SELECT DISTINCT ord_code
FROM Order_line
GROUP BY ord_code, art_code
HAVING COUNT(*) > 1;
```

or

```
SELECT DISTINCT P1.ord_code
FROM Order_line P1, Order_line P2
WHERE P1.ord_code = P2.ord_code AND P1.ord_line <> P2.ord_line AND
            P1.art_code = P2.art_code;
```

or

```
SELECT ord_code
FROM Order_line
GROUP BY ord_code
HAVING COUNT(art_code) > COUNT(DISTINCT art_code);
```

6)

```
CREATE ASSERTION R1
CHECK (NOT EXISTS (SELECT *
                        FROM Invoice F
                        WHERE (SELECT COUNT(*)
                                FROM Invoice_line LF
                                WHERE F.inv_code = LF.inv_code) = 0))
```

or

```
CREATE ASSERTION R1
CHECK (NOT EXISTS (SELECT *
                        FROM Invoice F
                        WHERE NOT EXISTS (SELECT *
                                        FROM Invoice_line LF
                                        WHERE F.inv_code = LF.inv_code)))
```

7)

```
SELECT C.name, SUM(F.amount)
FROM Invoice F, Customer C
WHERE F.cust_code = C.cust_code
GROUP BY C.cust_code, C.name
HAVING SUM(F.amount) >= ALL (SELECT SUM(F2.amount)
                                FROM Invoice F2
                                GROUP BY F2.cust_code)
```

8)
(a) The student should have chosen 5 among the following (except from DELETE from Order_line which was already given):

- INSERT into Order_line
- DELETE from Order_line
- UPDATE ord_code in Order_line
- UPDATE art_code in Order_line
- UPDATE quantity in Order_line
- UPDATE price in Article
- INSERT into Order
- UPDATE amount in Order

(b)

```
CREATE TRIGGER linea_order_DELETE
AFTER DELETE ON Order_line
FOR EACH ROW
DECLARE
        x NUMBER
BEGIN
        SELECT price INTO x FROM Article
        WHERE art_code=:OLD.art_code;

        UPDATE Order SET amount = amount – x * :OLD.quantity
        WHERE ord_code=:OLD.ord_code;
END ;
```