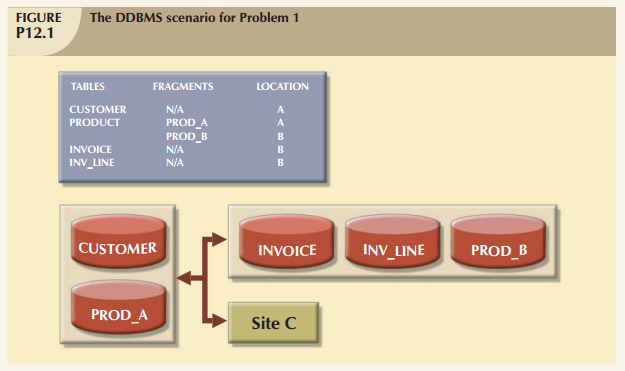
**44-560 Advanced Topics in Database Systems**

**Distributed Databases WS 01**

1. At the end of chapter 12, beginning on page 536, do problem 1, all parts. Note that in Figure P12.1, Site A is the site with tables CUSTOMER and PROD\_A; site B is the site with tables INVOICE, INV\_LINE, and PROD\_B.



Specify the minimum type(s) of operation(s) the database must support (remote request, remote transaction, distributed transaction, or distributed request) to perform the following operations:

**At site C:**

a. SELECT \* FROM CUSTOMER;

b. SELECT \* FROM INVOICE WHERE INV\_TOT > 1000;

c. SELECT \* FROM PRODUCT WHERE PROD\_ QOH < 10;

d. BEGIN WORK;

UPDATE CUSTOMER SET CUS\_BAL = CUS\_BAL + 100 WHERE CUS\_NUM = '10936';

INSERT INTO INVOICE(INV\_NUM, CUS\_NUM, INV\_DATE, INV\_TOTAL) VALUES ('986391', '10936', '15-FEB-2008', 100);

INSERT INTO LINE(INV\_NUM, PROD\_NUM, LINE\_PRICE) VALUES('986391', '1023', 100);

UPDATE PRODUCT SET PROD\_QOH = PROD\_ QOH –1 WHERE PROD\_NUM = '1023';

COMMIT WORK;

e. BEGIN WORK;

INSERT INTO CUSTOMER(CUS\_NUM, CUS\_NAME, CUS\_ADDRESS, CUS\_BAL) VALUES ('34210', 'Victor Ephanor', '123 Main St.', 0.00);

INSERT INTO INVOICE(INV\_NUM, CUS\_NUM, INV\_DATE, INV\_TOTAL) VALUES ('986434', '34210', '10-AUG-2007', 2.00);

COMMIT WORK;

**At site A**

f. SELECT CUS\_NUM,CUS\_NAME,INV\_TOTAL FROM CUSTOMER, INVOICE WHERE CUSTOMER.CUS\_NUM = INVOICE.CUS\_NUM;

g. SELECT \* FROM INVOICE WHERE INV\_TOTAL > 1000;

h. SELECT \* FROM PRODUCT WHERE PROD\_QOH < 10;

**At site B**

i. SELECT \* FROM CUSTOMER;

j. SELECT CUS\_NAME, INV\_TOTAL FROM CUSTOMER, INVOICE WHERE INV\_TOTAL > 1000 AND CUSTOMER.CUS\_NUM = INVOICE.CUS\_NUM;

k. SELECT \* FROM PRODUCT WHERE PROD\_QOH < 10;

1. This question uses the same distributed database as in problem 1. Tell whether each of the following represents a remote request, remote transaction, distributed transaction, or distributed request.
   1. At site C:

**UPDATE CUSTOMER**

**SET CUS\_BAL = 5000**

**WHERE CUS\_NUM = ‘1234’;**

* 1. At site C:

**BEGIN WORK;**

**UPDATE CUSTOMER**

**SET CUS\_BAL = 5000**

**WHERE CUS\_NUM = ‘1234’;**

**INSERT INTO CUSTOMER(CUS\_NUM, CUS\_NAME, CUS\_ADDRESS, CUST\_BAL)**

**VALUES(‘1234’, ‘Amy Smith’, ‘234 Avenue K’, 1000.00);**

**COMMIT WORK;**

* 1. At site C:

**BEGIN WORK;**

**UPDATE CUSTOMER**

**SET CUS\_BAL = 5000**

**WHERE CUS\_NUM = ‘1234’;**

**INSERT INTO CUSTOMER(CUS\_NUM, CUS\_NAME, CUS\_ADDRESS, CUST\_BAL)**

**VALUES(‘1234’, ‘Amy Smith’, ‘234 Avenue K’, 1000.00);**

**INSERT INTO INVOICE(INV\_NUM, CUS\_NUM, INV\_DATE, INV\_TOTAL)**

**VALUES(‘111222333’, ‘1234’, ‘October 10, 2009’, 157.25);**

**COMMIT WORK;**

1. Suppose we have the following student data stored in a university database for a multi-campus university.

**STUDENT**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **LName** | **FName** | **Campus** | **Major** | **GPA** | **HealthPlan** | **FinAidPlan** | **AcctBal** |
| **1** | **Brown** | **Alan** | **Park** | **Math** | **3.8** | **Basic** | **A** | **1,234.56** |
| **2** | **Smith** | **Sue** | **Bayside** | **Math** | **3.9** | **BasicPlus** | **A** | **543.67** |
| **3** | **Carter** | **Kay** | **Bayside** | **CS** | **3.8** | **Basic** | **B** | **5,234.00** |
| **4** | **Thomas** | **Bill** | **Park** | **Math** | **3.0** | **Basic** | **A** | **333.47** |
| **5** | **Anders** | **Andy** | **Park** | **CS** | **2.5** | **Basic** | **C** | **123.45** |

We will store this database in fragments, using mixed fragmentation.

* 1. Begin by fragmenting horizontally by campus. Call the fragments **STUDENT\_PARK** and **STUDENT\_BAYSIDE**, and show the contents of each fragment.
  2. Next, you will further fragment each fragment using vertical fragmentation. At each campus, the Registrar’s will have the academic information about each student, including name of student, campus, major, and gpa. Health Services will have information about health plans, and Cashiering will have information about financial aid plans and account balances. Name each of the new fragments using the existing fragment name, followed by an underscore, followed by **REGISTRAR**, **HEALTH**, or **CASHIERING**. Note that only the registrar will store the name of the student, but *all* fragments must contain the student’s id.