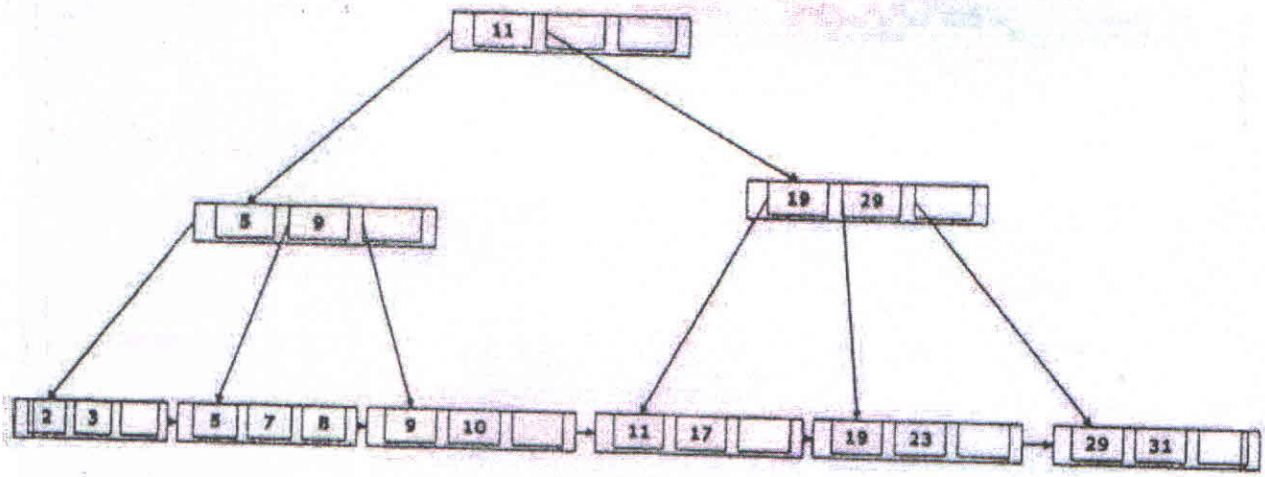
Sardar Patel Institute of Technology
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058, India
(Autonomous College Affiliated to University of Mumbai)
Mid Semester Examination
September 2018

Max. Marks: 20 **Duration: 60 mins**
Class: T.E. **Semester: V**
Course Code: IT53 **Branch: Information Technology**
Name of the Course: Advanced Database Systems

Synoptic

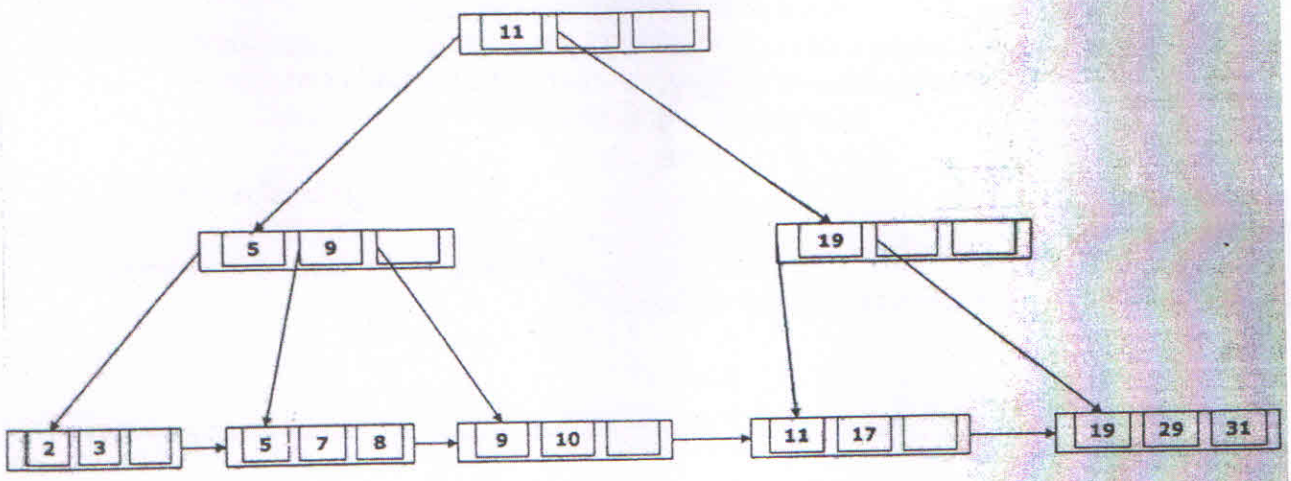
Q.1. Analyze the following B+ tree and illustrate the delete operation and show the B+ tree after each deletion for following numbers are deleted :

23,19,17,10,11

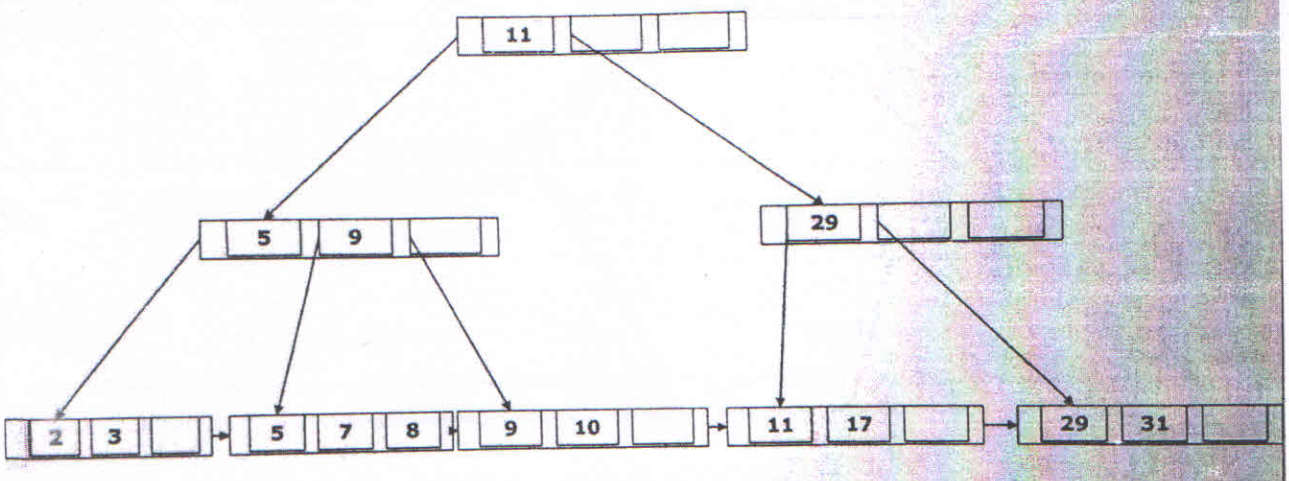


Ans: **Each deletion and tree 1 mark**

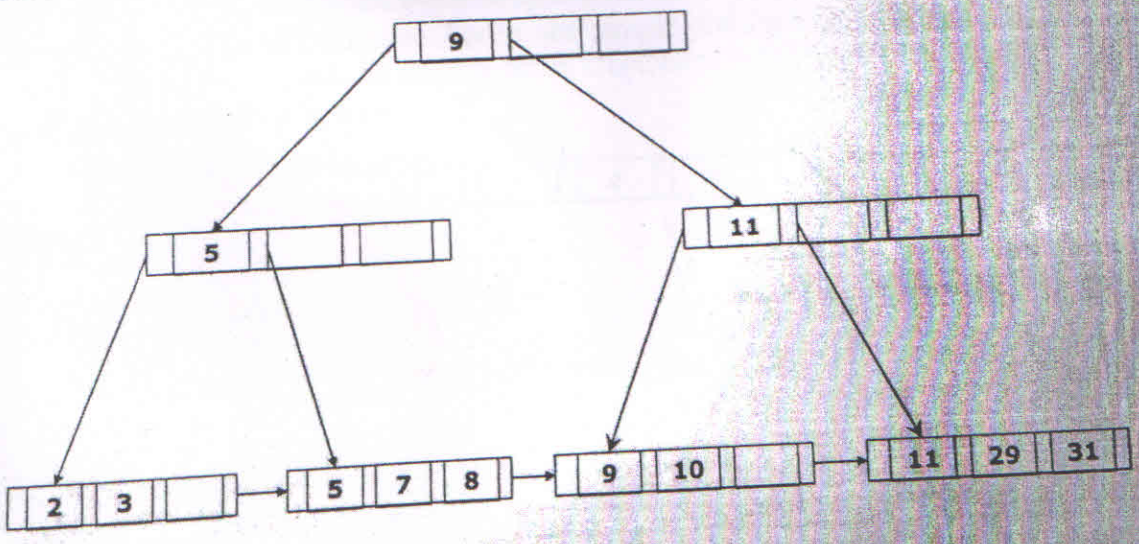
Delete 23



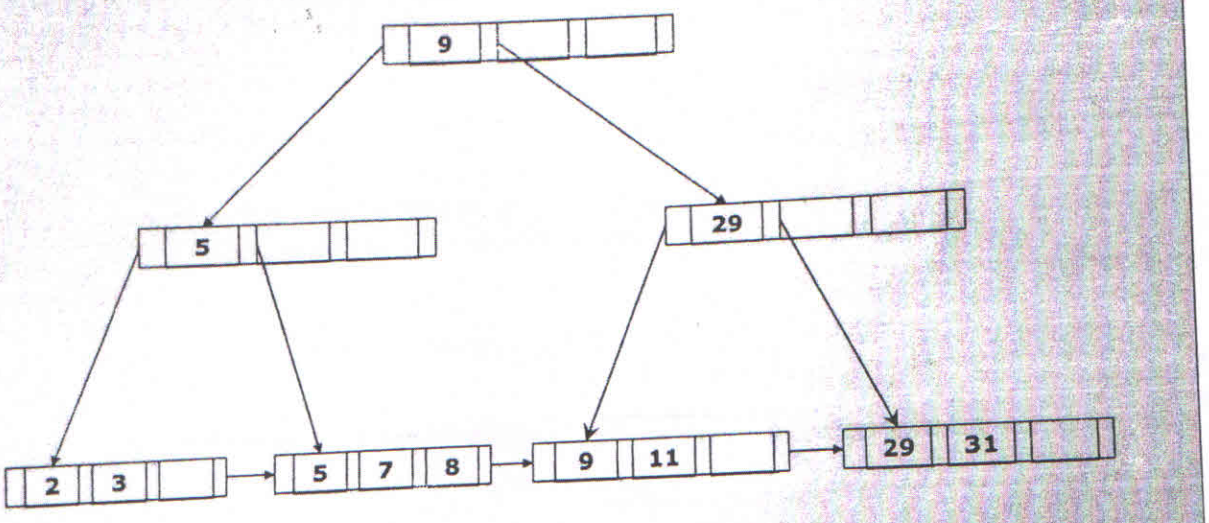
Delete 19



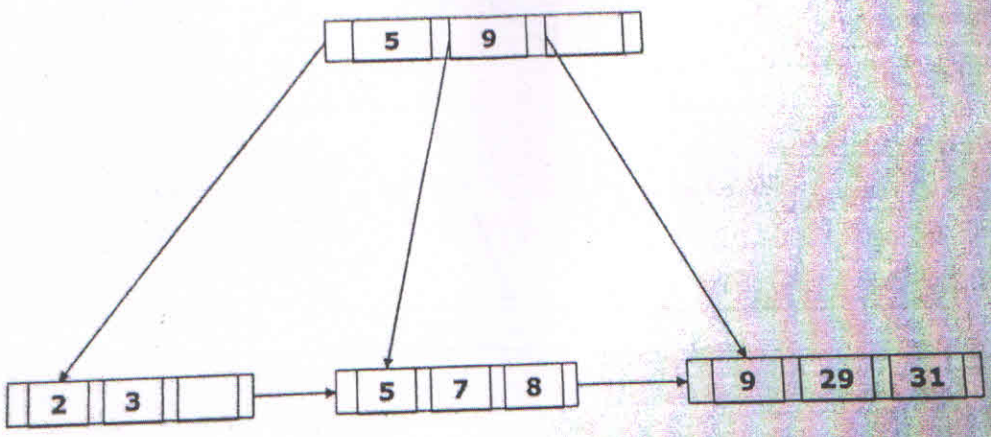
Delete 17



Delete 10



Delete 11



Q.2. Consider the following relational schema and set of applications that are frequently accessing the relation ;

Department(DeptNo, DeptName, College , Phone)

Application1: Find the department details of SPIT College.

Application2: Find the phone of IT department

Create the horizontal fragmentation using simple predicates and minterm predicates according to the requirement and check for the correctness of the fragments.

Ans:

lution:

imple predicates:

From the applications, we can identify the simple predicates. The first application access the department table using the school name. hence, the simple predicate is College = 'SPIT'. The second application access the data on the condition DeptName = 'IT'. Hence, our set of simple predicates Pr can be written as follows;

Pr = { p1: College = 'SPIT', p2: DeptName = 'IT'}

1 mark

Min-term predicates:

Min-term predicates can be derived from set of simple predicates by ANDing and NEGATing all the simple predicates as follows;

m1 = { College = 'SPIT' ^ DeptName = 'IT'}

m2 = { College = 'SPIT' ¬DeptName = 'IT '}

m3 = { ¬(College = 'SPIT ') ^ DeptName = 'IT'}

m4 = { ¬(College = 'SPIT') ^ ¬(DeptName = 'IT')}

We have negated each simple predicate so as not to miss any records from the tables. For example, when we mention College = 'SPIT' it simplicitly means that there are other college other that SPIT. And the condition college = 'SPIT' includes SPIT college and the negations of this i.e., College <> 'SPIT' will include all the other colleges.

1 mark

Primary Horizontal Fragmentation:

The table DEPARTMENT can be horizontally fragmented using the Primary Horizontal Fragmentation technique using these min-term predicates m1, m2, m3 and m4. For example, the fragment 1 stores the data as per the min-term predicate m1 using the following query;

SELECT * FROM Department WHERE College = 'SPIT' AND DeptName = 'IT ';

At the end of fragmentation, we have 4 fragments of Department table viz., $DEPT_1$, $DEPT_2$, $DEPT_3$ and $DEPT_4$.

After fragmenting a table, the very next step is to check whether the fragments are correct or not. This could be verified using the following correctness properties;

1 mark

Completeness – Each record of table DEPARTMENT should be found in any one of the fragments $DEPT_1$, $DEPT_2$, $DEPT_3$ and $DEPT_4$. As our simple predicates are complete and minimal, we can say that fragments are complete.

Reconstruction – We must be able to reconstruct DEPARTMENT from the fragments $DEPT_1$, $DEPT_2$, $DEPT_3$ and $DEPT_4$. The following relational algebra operation on fragments will get us DEPARTMENT;

$$DEPARTMENT = DEPT_1 \cup DEPT_2 \cup DEPT_3 \cup DEPT_4$$

Dis-jointness – The result of the intersect operation between fragments should give me a emplty set as result;

$$DEPT_1 \cup DEPT_2 \cup DEPT_3 \cup DEPT_4 = \emptyset$$

All these three properties are verified. Hence, our primary horizontal fragmentation is correct.

2 marks

Q3. Consider airline booking system and if you want to implement the following schema

Customer (CustomerId, Title,FirstName, LastName,Gender, Age,Address, Email)

Flight(FlightNo, DepartureDate, DepartureTime, ArrivalDate, ArrivalTime)

Seat(FightNo,CustimerId,DepartureDate,SeatNo,Class)

Ticket(SerialNo,FlightNo,DepartureDate,SeatNo,CustomerId)

Choose and justify any four transparencies are used for designing the distributed database system for airline booking systems

Ans: Indentifying 4 transparencies **1 mark**

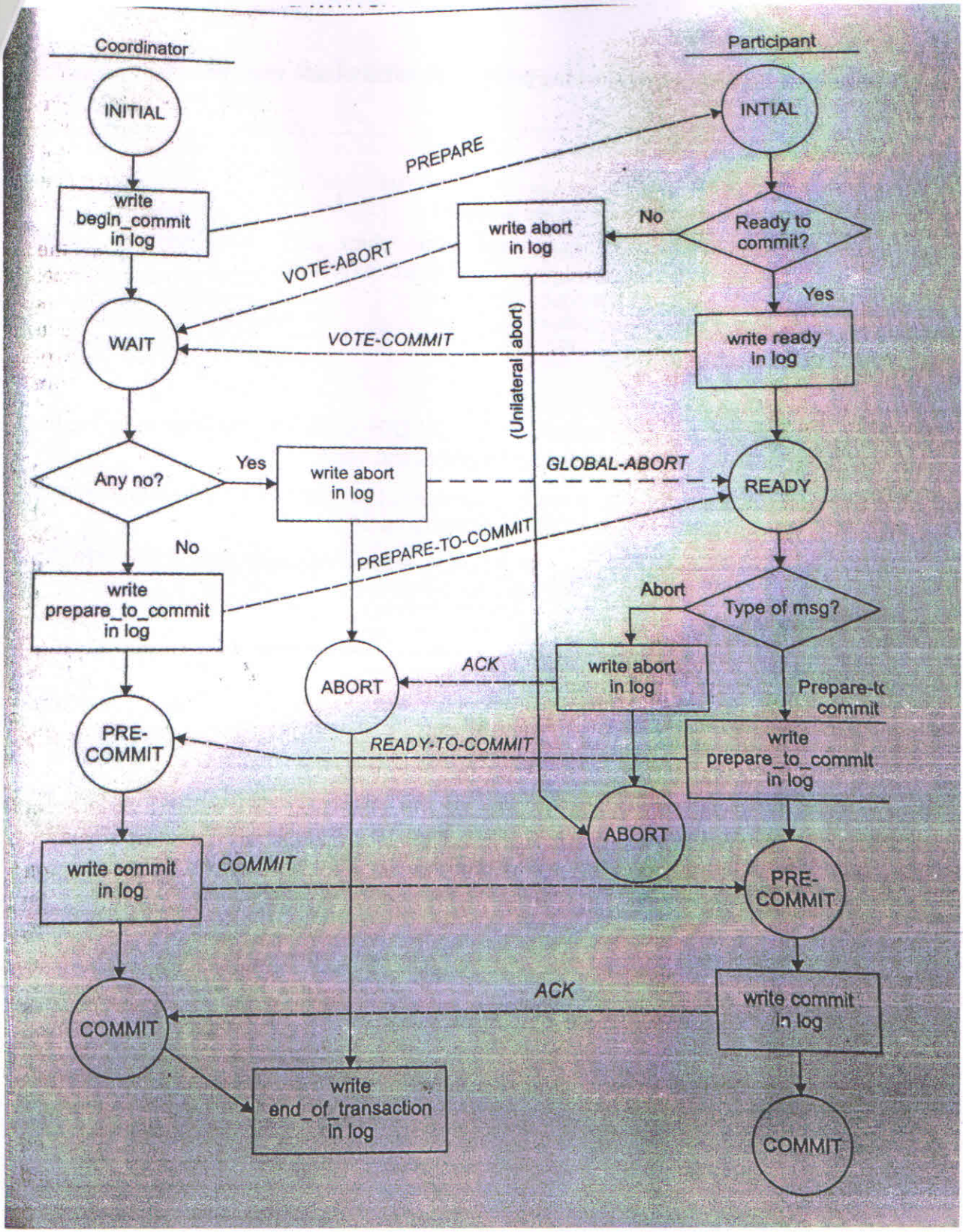
Justification of 4 with respect to above case study **4 marks**

OR

Q.3. Three Phase Commit protocol is designed as nonblocking protocol with the help of diagram show the 3PC protocol actions and describe in brief three phases of 3PC.

Ans:

Diagram **2 marks**


Coordinator
Participant
INITIAL
INTIAL
PREPARE
write begin_commit in log
write abort in log
No
Ready to commit?
VOTE-ABORT
Yes
WAIT
VOTE-COMMIT
write ready in log
(Unilateral abort)
Any no?
Yes
write abort in log
GLOBAL-ABORT
READY
No
PREPARE-TO-COMMIT
write prepare_to_commit in log
Abort
Type of msg?
ABORT
ACK
write abort in log
Prepare-to commit
PRE-COMMIT
READY-TO-COMMIT
write prepare_to_commit in log
ABORT
write commit in log
COMMIT
PRE-COMMIT
ACK
COMMIT
write commit in log
write end_of_transaction in log
COMMIT

Description of 3 phases **3 marks**

3PC is a protocol that eliminates this blocking problem on certain basic requirements;

- No network partitioning
- At least one site must be available
- At most K simultaneous site failures are accepted

2PC has two phases namely voting phase and decision phase. 3PC introduces pre-commit phase (serves as a buffer phase) as the third phase. 3PC works as follows;

Phase 1 (WAIT/VOTING):

Transaction Coordinator (TC) of the transaction writes BEGIN_COMMIT message in its log file and sends PREPARE message to all the participating sites and waits.

Upon receiving this message, if a site is ready to commit, then the site's transaction manager (TM) writes READY in its log and send VOTE_COMMIT to TC.

If any site is not ready to commit, it writes ABORT in its log and responds with VOTE_ABORT to the TC.

Phase 2 (PRE-COMMIT):

If TC received VOTE_COMMIT from all the participating sites, then it writes PREPARE_TO_COMMIT in its log and sends PREPARE_TO_COMMIT message to all the participating sites.

On the other hand, if TC receives any one VOTE_ABORT message, it writes ABORT in its log and sends GLOBAL_ABORT to all the participating sites and also writes END_OF_TRANSACTION message in its log.

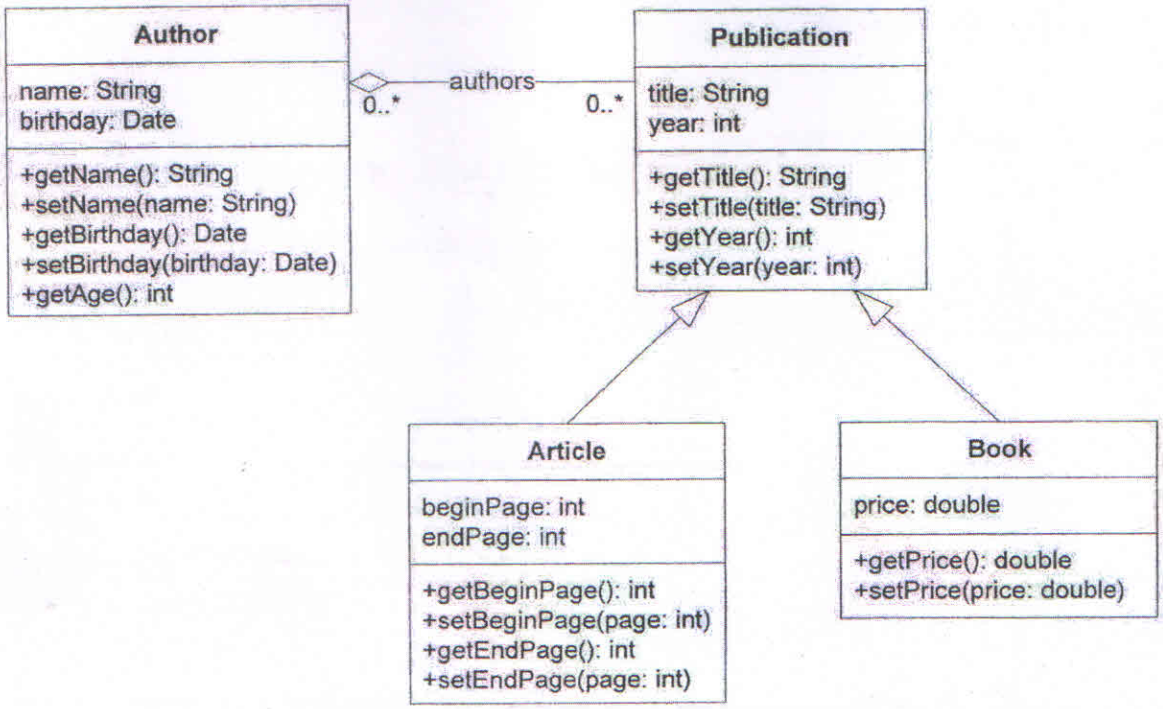
On receiving the message PREPARE_TO_COMMIT, the TM of participating sites write PREPARE_TO_COMMIT in their log and respond with READY_TO_COMMIT message to the TC.

If they receive GLOBAL_ABORT message, then TM of the sites write ABORT in their logs and acknowledge the abort. Also, they abort that particular transaction locally.

Phase 3 (COMMIT/DECIDING):

If all responses are READY_TO_COMMIT, then TC writes COMMIT in its log and send GLOBAL_COMMIT message to all the participating sites' TMs. The TM of those sites then writes COMMIT in their log and sends an acknowledgement to the TC. Then, TC writes END_OF_TRANSACTION in its log.

Q.4. Develop a ODL schema for the following databse.



Ans :

Author class 2 marks
Remaining classes 1 mark each

Class Author (extent Authors) {
Attribute string name;
Attribute date birthday;
Relationship set<Publication> authors inverse Publication::authored_by;
Integer get_age();
};
Class Publication (extentPublications) {
Attribute string title;
Attribute integer year;
Relationship list<Author> authored_by inverse Author::authors;
};
Class Article extends Publication (extent Articles) {
Attribute unsigned short begin_page;
Attribute unsigned short end_page;
};

Class Book extends Publication (extent Books) {
Attribute double price;
};