Τ. Ε. Ι. ΠΕΛΟΠΟΝΝΗΣΟΥ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΤΕΧΝΙΚΕΣ ΑΝΑΚΑΜΨΗΣ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ

Φοιτητής: ΣΕΡΑΣΚΕΡΗ ΔΕΣΠΟΙΝΑ

Σπάρτη 2015

ΠΕΡΙΛΗΨΗ

Στην παρούσα πτυχιακή εργασία παρουσιάζεται ο όρος συναλλαγή στο γνωστικό αντικείμενο των Βάσεων Δεδομένων και γίνεται μια προσπάθεια μελέτης των χαρακτηριστικών των συναλλαγών καθώς και των τεχνικών ανάκαμψης που μπορούν να εφαρμοσθούν σε μια βάση δεδομένων. Στη συνέχεια δίνεται έμφαση στα σφάλματα ταυτοχρονισμού εκτέλεσης των συναλλαγών που προκύπτουν και παρουσιάζονται τα πρωτόκολλα κλειδώματος που αποτελούν λύση αυτών. Επιπλέον αναλύεται ο αλγόριθμος ARIES και οι μελλοντικές εκδόσεις του που αφορούν τεχνικές επαναφοράς συναλλαγών μετά από κατάρρευση συστήματος.

Στο 1^{ο} κεφάλαιο δίνεται μια εκτενής περιγραφή της συναλλαγής τόσο στην περιοχή των Βάσεων Δεδομένων όσο και σε διάφορους τομείς της καθημερινής ζωής. Παρουσιάζονται οι ιδιότητες που απαιτείται να διατηρεί μια συναλλαγή ως προς τη διασφάλιση της ακεραιότητας των δεδομένων σύμφωνα με το πρότυπο ACID. Επιπλέον, περιγράφεται το μοντέλο του χρονοπρογράμματος και των κατηγοριών που το απαρτίζουν. Συγκεκριμένα καταγράφονται κατηγορίες χρονοπρογραμμάτων και δίνονται λύσεις σε προβλήματα που ανακύπτουν από συγκρούσεις συναλλαγών.

Στο 2ο κεφάλαιο περιγράφεται η ταυτόχρονη επιτρεπόμενη εκτέλεση πολλών συναλλαγών και εξετάζονται οι επιδόσεις των ΣΔΒΔ ώστε να μη δημιουργούνται προβλήματα. Επίσης, δίνονται παραδείγματα αυτών και εκμαιεύονται αποτελέσματα, τα οποία οδηγούν στην δημιουργία πρωτοκόλλων που καθορίζουν τον τρόπο λειτουργίας των ΣΔΒΔ ως προς τη σειριακή και εναλλασσόμενη εκτέλεση συναλλαγών.

Στο 3^{ο} κεφάλαιο παρουσιάζεται το πρωτόκολλο κλειδώματος 2PL και μελετώνται σενάρια εφαρμογής του προαναφερθέντος πρωτοκόλλου. Επίσης, περιγράφονται κατηγορίες του πρωτοκόλλου και ερευνάται η έννοια του αδιέξοδου και η κατάργηση του σε περιπτώσεις εμφάνισής του.

Στο 4^{ο} κεφάλαιο γίνεται εκτενής αναφορά στον αλγόριθμο επαναφοράς συστήματος ARIES και στις εκδόσεις που προτάθηκαν βασιζόμενες στην αρχική ιδέα. Επίσης συγκρίνονται οι μεταγενέστερες εκδόσεις του αλγορίθμου και εντοπίζονται πλεονεκτήματα και μειονεκτήματα αυτών.

Τέλος παρουσιάζονται τα συμπεράσματα που προέκυψαν από τη βιβλιογραφική επισκόπηση και αποτιμάται η εφαρμογή του αλγορίθμου ARIES σε ΣΔΒΔ που κατέρρευσαν μετά από αποτυχία του υλικού είτε του λογισμικού. Επιπλέον προτείνονται μελλοντικά ερευνητικά ερωτήματα στον τομέα των Κατανεμημένων Συστημάτων Διαχείρισης Βάσεων Δεδομένων.

ABSTRACT

This thesis presents the term transaction in the subject of Database and it is being considered the characteristics of the transactions and the recovery techniques that can be applied to a database. Then the emphasis on concurrency errors trade execution obtained and presented by locking protocols which form such a solution. Additional analyzes the ARIES algorithm and future versions of related transaction recovery techniques after a system crash.

The 1st chapter gives a comprehensive description of the transaction in both the Database area and in different areas of daily life. Featured properties required to maintain a transaction in ensuring data integrity according to the standard ACID. Furthermore, the model describes the timer program and its component categories. Specifically listed categories schedules and provided solutions to problems arising from trade conflicts.

The 2nd section describes the allowable running multiple simultaneous transactions and examined the performance of the DBMS so as to avoid problems. They also provide examples of these and elicited results, which lead to the creation of protocols that define how the DBMS on the serial and alternating execution of transactions.

The 3rd chapter presents the lock 2PL protocol and studied application scenarios of the said Protocol. Also we described categories of the Protocol and investigated the concept of deadlock and the abolition in cases of appearance.

The 4th chapter is an extensive reference to the system restore algorithm and we proposed ARIES versions based on the original idea. Moreover we compare the later versions of the algorithm and we identify advantages and disadvantages of these.

Finally we presented the conclusions drawn from the literature review and evaluating the implementation of the algorithm in ARIES in DBMS collapsed after failure of hardware or software. Additional we proposed future research questions in the field of Distributed Database Management Systems.

ΕΥΧΑΡΙΣΤΙΕΣ

Η ολοκλήρωση αυτής της πτυχιακής υλοποιήθηκε με την υποστήριξη κάποιον ανθρώπων στους οποίους θα ήθελα να εκφράσω τις θερμότερες ευχαριστίες μου. Πρώτα από όλους θα ήθελα να ευχαριστήσω την καθηγήτρια μου κ.Σαλτάρη Γεωργία για την βοήθεια και υποστήριξη της ώστε να ολοκληρωθεί με επιτυχία η εργασία μου. Ευχαριστώ τους γονείς μου για την στήριξη τους όλα αυτά τα χρόνια των σπουδών μου που με βοήθησαν να φτάσω ως το τέλος.

Σερασκέρη Δέσποινα

Νοέμβριος 2015

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΛΗΨΗ **III**

ABSTRACT **V**

ΕΥΧΑΡΙΣΤΙΕΣ **VII**

ΠΕΡΙΕΧΟΜΕΝΑ **VIII**

ΕΙΣΑΓΩΓΗ **XI**

ΔΙΑΧΕΙΡΙΣΗ ΣΥΝΑΛΛΑΓΩΝ **13**

1.1 Εισαγωγή **13**

1.2 Τι είναι συναλλαγή **15**
1.2.1 Σύνταξη συναλλαγής σε γλώσσα SQL 16
1.2.1.1 Παράδειγμα συναλλαγής 17

1.3 Ιδιότητες συναλλαγών **19**
1.3.1 Ατομικότητα 20
1.3.2 Συνέπεια 20
1.3.3 Απομόνωση 21
1.3.4 Μονιμότητα 23

1.4 Καταστάσεις συναλλαγών **24**

1.5 Χρονοπρογράμματα **26**
1.5.1 Τι είναι 26
1.5.2 Σειριακό χρονοπρόγραμμα 28
1.5.3 Σειριοποιήσιμο χρονοπρόγραμμα 29

1.6 Συγκρούσεις **29**
1.6.1 Ισοδυναμία συγκρούσεων 30
1.6.2 Σειριοποιησιμότητα συγκρούσεων 31
1.6.3 Ισοδυναμία όψεως 31
1.6.4 Σειριοποιησιμότητα όψεως 32
1.6.5 Ανακτησιμότητα 33
1.6.5.1 Μη – ανακτήσιμο 33
1.6.5.2 Αποφυγή διαδοχικών abort 34
1.6.6 Αυστηρότητα 35

ΤΑΥΤΟΧΡΟΝΗ ΕΚΤΕΛΕΣΗ ΠΟΛΛΩΝ ΣΥΝΑΛΛΑΓΩΝ **36**

2.1 Εισαγωγή **36**

2.2 Πλεονεκτήματα από την ταυτόχρονη εκτέλεση συναλλαγών **37**

2.3 Προβλήματα από την εναλλασσόμενη εκτέλεση συναλλαγών **38**
2.3.1 1η περίπτωση - Το πρόβλημα της χαμένης ενημέρωσης (lost update problem) 38
2.3.2 2η περίπτωση - Το πρόβλημα της λανθασμένης ανάγνωσης (dirty read) 39
2.3.3 3η περίπτωση - Το πρόβλημα της παραγωγής συγκεντρωτικών στοιχείων 39

2.4 Συγκρούσεις προς αποφυγή **40**
2.4.1 Ανάγνωση μη ολοκληρωμένων δεδομένων (Συγκρούσεις WR) 41
2.4.2 Μη επαναλαμβανόμενες αναγνώσεις (Συγκρούσεις RW) 42
2.4.3 Επανεγγραφή μη-ολοκληρωμένων δεδομένων (Συγκρούσεις WW) 43

ΕΛΕΓΧΟΣ ΤΑΥΤΟΧΡΟΝΙΣΜΟΥ ΜΕ ΚΛΕΙΔΩΜΑΤΑ **45**

3.1 Εισαγωγή **45**

3.2 Μηχανισμοί κλειδώματος **45**

3.3 Πρωτόκολλα κλειδώματος - Αυστηρό 2PL **48**

3.4 Πρωτόκολλο κλειδώματος βασισμένο σε γράφημα **51**

3.5 Διαχείριση αδιεξόδου **53**

3.6 Αποφυγή αδιεξόδου **54**

3.7 Αναγνώριση και κατάργηση αδιεξόδου **55**

ΕΠΑΝΑΦΟΡΑ ΔΕΔΟΜΕΝΩΝ ΜΕΤΑ ΑΠΟ ΚΑΤΑΡΡΕΥΣΗ ΣΥΣΤΗΜΑΤΟΣ **59**

4.1 Εισαγωγή **59**

4.2 Απαιτήσεις επανάκτησης **60**

4.3 Μέθοδοι επαναφοράς **62**

4.4 Ο αλγόριθμος ARIES **63**
4.4.1 Πρωτόκολλο WAL 64
4.4.2 Καταγραφή 65
4.4.3 Επαναφορά 67
4.4.3.1 Ανάλυση 67
4.4.3.2 Επανάληψη 68
4.4.3.3 Αναίρεση 69
4.4.4 Σημεία ελέγχου 71

4.5 Μελλοντικές εκδόσεις του αλγορίθμου ARIES **72**
4.5.1 ARIES-RRH 72
4.5.2 ARIES/NT 73

ΕΥΡΕΤΗΡΙΟ ΕΙΚΟΝΩΝ

Εικόνα 1-Αρχιτεκτονική μιας εφαρμογής ΒΔ 15
Εικόνα 2-Τμήμα Κώδικα Συναλλαγής 16
Εικόνα 3 - Εκτέλεση Συναλλαγών 22
Εικόνα 4 - Καταστάσεις Συναλλαγών 25
Εικόνα 5- Παράδειγμα χρονοπρογράμματος 27
Εικόνα 6-Παράδειγμα σειριακού χρονοπρογράμματος 29
Εικόνα 7 - Παράδειγμα χρονοπρογράμματος με συγκρούσεις 31
Εικόνα 8 - Παράδειγμα χρονοπρογράμματος με σειριοποιησιμότητα συγκρούσεων 32
Εικόνα 9 - Παράδειγμα χρονοπρογράμματος με σειριοποιησιμότητα συγκρούσεων 32
Εικόνα 10 - Παράδειγμα ανακτησιμότητας συγκρούσεων 33
Εικόνα 11 - Παράδειγμα μη ανακτήσιμου προγράμματος 34
Εικόνα 12 - Παράδειγμα χρονοπρογραμμάτων abort 34
Εικόνα 13 - Παράδειγμα ανακτήσιμου προγράμματος με αποφυγή abort 35
Εικόνα 14 - Προβλήματα που προκύπτουν από την ταυτόχρονη εκτέλεση συναλλαγών 40
Εικόνα 15 - Χρονοπρόγραμμα με συγκρούσεις WR 42
Εικόνα 16 - Παράδειγμα χρήσης μηχανισμού κλειδώματος 48
Εικόνα 17 - Χρονοπρόγραμμα που περιγράφει το Αυστηρό πρωτόκολλο 2PL 50
Εικόνα 18 - Εναλλαγή των ενεργειών με το Αυστηρό 2PL πρωτόκολλο 51
Εικόνα 19 - Παράδειγμα εφαρμογής του δενδρικού πρωτοκόλλου κλειδώματος 53
Εικόνα 20 - Παράδειγμα γραφήματος αναμονής χωρίς κύκλο και με κύκλο 56
Εικόνα 21 - Παράδειγμα επαναφοράς 60
Εικόνα 22 - Οι διακεκομμένες γραμμές μπορεί να αναπαριστούν λειτουργίες που θα καθυστερήσουν ή να αγνοηθούν 66
Εικόνα 23 - Παράδειγμα εφαρμογής του αλγορίθμου 70

ΕΙΣΑΓΩΓΗ

Στο 1^{ο} κεφάλαιο δίνεται μια εκτενής περιγραφή της συναλλαγής τόσο στην περιοχή των Βάσεων Δεδομένων όσο και σε διάφορους τομείς της καθημερινής ζωής. Παρουσιάζονται οι ιδιότητες που απαιτείται να διατηρεί μια συναλλαγή ως προς τη διασφάλιση της ακεραιότητας των δεδομένων σύμφωνα με το πρότυπο ACID. Επιπλέον, περιγράφεται το μοντέλο του χρονοπρογράμματος και των κατηγοριών που το απαρτίζουν. Συγκεκριμένα καταγράφονται κατηγορίες χρονοπρογραμμάτων και δίνονται λύσεις σε προβλήματα που ανακύπτουν από συγκρούσεις συναλλαγών.

Στο 2ο κεφάλαιο περιγράφεται η ταυτόχρονη επιτρεπόμενη εκτέλεση πολλών συναλλαγών και εξετάζονται οι επιδόσεις των ΣΔΒΔ ώστε να μη δημιουργούνται προβλήματα. Επίσης, δίνονται παραδείγματα αυτών και εκμαιεύονται αποτελέσματα, τα οποία οδηγούν στην δημιουργία πρωτοκόλλων που καθορίζουν τον τρόπο λειτουργίας των ΣΔΒΔ ως προς τη σειριακή και εναλλασσόμενη εκτέλεση συναλλαγών.

Στο 3^{ο} κεφάλαιο παρουσιάζεται το πρωτόκολλο κλειδώματος 2PL και μελετώνται σενάρια εφαρμογής του προαναφερθέντος πρωτοκόλλου. Επίσης, περιγράφονται κατηγορίες του πρωτοκόλλου και ερευνάται η έννοια του αδιέξοδου και η κατάργηση του σε περιπτώσεις εμφάνισής του.

Στο 4^{ο} κεφάλαιο γίνεται εκτενής αναφορά στον αλγόριθμο επαναφοράς συστήματος ARIES και στις εκδόσεις που προτάθηκαν βασιζόμενες στην αρχική ιδέα.

Τέλος παρουσιάζονται συμπεράσματα που προέκυψαν από τη βιβλιογραφική επισκόπηση και προτείνονται μελλοντικά ερευνητικά ερωτήματα.

1 ο ΚΕΦΑΛΑΙΟ

ΔΙΑΧΕΙΡΙΣΗ ΣΥΝΑΛΛΑΓΩΝ

1.1 Εισαγωγή

Στην καθημερινή ζωή, οι άνθρωποι διεκπεραιώνουν ή διεξάγουν διαφορετικά είδη επιχειρηματικών συναλλαγών, όπως: αγορά προϊόντων, κράτηση ταξιδιών, αλλαγή ή ακύρωση παραγγελιών, αγορά εισιτηρίων για συναυλίες, πληρωμή ενοικίου και λογαριασμών ηλεκτρικού ρεύματος, τιμολόγια ασφάλισης/ασφαλιστικής κάλυψης/ασφαλιστήρια, κλπ. Οι συναλλαγές δεν αφορούν μόνο σε υπολογιστές, φυσικά. Κάθε είδος της ανθρώπινης δραστηριότητας που περιλαμβάνει μία λογική μονάδα εργασίας, που (σημαίνει ότι) είτε πρέπει να εκτελεστεί στο σύνολό της ή να ακυρωθεί στο σύνολό της αποτελεί μια συναλλαγή.

Σχεδόν όλα τα πληροφοριακά συστήματα χρησιμοποιούν τις υπηρεσίες κάποιου Συστήματος Διαχείρισης Βάσεων Δεδομένων (ΣΔΒΔ) για την αποθήκευση και ανάκτηση δεδομένων. Τα σύγχρονα ΣΔΒΔ είναι τεχνικά εξελιγμένα εξασφαλίζοντας την ακεραιότητα των δεδομένων στις βάσεις δεδομένων τους και παρέχουν γρήγορη πρόσβαση στα δεδομένα ακόμα και από πολλούς ταυτόχρονους χρήστες. Επίσης, προσφέρουν αξιόπιστες υπηρεσίες στις εφαρμογές για τη διαχείριση της ανθεκτικότητας των δεδομένων (persistency of data), αλλά μόνο αν οι εφαρμογές χρησιμοποιούν αυτές τις αξιόπιστες υπηρεσίες κατάλληλα. Αυτό εξασφαλίζεται αν

υλοποιηθούν τα τμήματα της αρχιτεκτονικής λογισμικού για την πρόσβαση στα δεδομένα χρησιμοποιώντας συναλλαγές βάσεων δεδομένων.

Η ακατάλληλη χρήση των συναλλαγών στις εφαρμογές λογισμικού, μπορεί να οδηγήσει

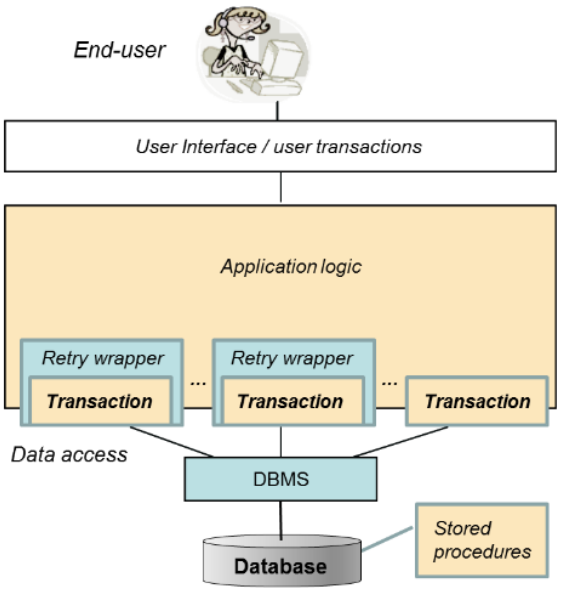
- στο να χαθούν παραγγελίες ή πληρωμές πελατών και αποστολές προϊόντων (στην περίπτωση ενός e-shop),
- σε αποτυχίες ή διπλοκρατήσεις στην κράτηση θέσεων για τρένα ή αεροπλάνα,
- σε χαμένες καταχωρήσεις κλήσεων σε κέντρα αντιμετώπισης καταστάσεων έκτακτης ανάγκης, κτλ.

Οι συναλλαγές, όσον αφορά τη διαχείριση του περιεχομένου μίας βάσης δεδομένων, είναι τμήματα (μονάδες) εργασιών πρόσβασης στα δεδομένα που έχουν τη δυνατότητα (σε περίπτωση αποτυχίας) να επανέλθουν στην αρχική τους κατάσταση.

Τέτοια περιστατικά συμβαίνουν συχνά, αλλά συνήθως οι υπεύθυνοι τα αποκρύπτουν από το κοινό. Επίσης, περιλαμβάνουν τρόπους επαναφοράς όλης της βάσης δεδομένων σε περίπτωση κατάρρευσης/πτώσης του συστήματος. Τέλος, προσφέρουν βασικές γνώσεις για τη διαχείριση του ταυτοχρονισμού σε περιβάλλοντα πολυχρηστίας.

Η Εικόνα 1, που ακολουθεί, παρουσιάζει μία απλουστευμένη άποψη της αρχιτεκτονικής μίας κλασικής εφαρμογής Βάσεων Δεδομένων, τοποθετώντας τις συναλλαγές βάσεων δεδομένων σε ξεχωριστό επίπεδο από το επίπεδο διεπαφής χρήστη. Από την οπτική γωνία του τελικού χρήστη, η εφαρμογή εξυπηρετεί τις περιπτώσεις χρήσης υλοποιώντας τες ως συναλλαγές χρηστών. Μία απλή συναλλαγή χρήστη μπορεί να περιλαμβάνει πολλαπλές SQL συναλλαγές, κάποιες από τις οποίες περιλαμβάνουν συναλλαγές ανάκτησης,

και συνήθως η τελευταία συναλλαγή στη σειρά ενημερώνει τα περιεχόμενα της βάσης δεδομένων. Οι ρουτίνες επανεκτέλεσης αποτελούν το μέσο για την εφαρμογή προγραμματιστικών ενεργειών επανεκτέλεσης, στην περίπτωση αποτυχίας του ταυτοχρονισμού στις SQL συναλλαγές.



Εικόνα 1-Αρχιτεκτονική μιας εφαρμογής ΒΔ

Το σύνολο των λειτουργιών, που αποτελούν μία μόνο λειτουργική μονάδα, ονομάζονται **συναλλαγές**. Μια Βάση Δεδομένων πρέπει να εξασφαλίζει τη σωστή εκτέλεση των συναλλαγών, ανεξάρτητα από τυχόν προβλήματα που μπορεί να παρουσιαστούν κατά την εκτέλεση, δηλαδή να εκτελείται είτε ολόκληρη η συναλλαγή είτε καθόλου.

1.2 Τι είναι συναλλαγή

Συναλλαγή είναι μια κινητή μονάδα εκτέλεσης ενός προγράμματος που προσπελαύνει και πιθανόν ενημερώνει διάφορα δεδομένα. Συνήθως, μια συναλλαγή ξεκινά από ένα πρόγραμμα χρήστη, που είναι γραμμένο σε μια

γλώσσα υψηλού επιπέδου ή γλώσσα προγραμματισμού (για παράδειγμα, SQL ή Java), όπου χωρίζεται από εντολές (ή κλήσεις συναρτήσεων) της μορφής begin transaction (έναρξη συναλλαγής) και end transaction (λήξη συναλλαγής). Η συναλλαγή αποτελείται από όλες τις λειτουργίες που εκτελούνται μεταξύ των begin_transaction και end_transaction.

1.2.1 Σύνταξη συναλλαγής σε γλώσσα SQL

Για να περιγράψουμε μια συναλλαγή με τη χρήση της γλώσσας SQL (Structured Query Language)χρησιμοποιούμε το παρακάτω τμήμα κώδικα:

```
TRANSACTION Ον_Συναλλαγής
WHENEVER {ERROR | συνθήκη_λάθους} ROLLBACK
                    Εντολές
                    Insert, Delete
                    Update κ.λπ.
COMMIT
END;
```

Εικόνα 2-Τμήμα Κώδικα Συναλλαγής

Στην πρώτη γραμμή έχουμε το όνομα της συναλλαγής για να αναφερόμαστε αργότερα σ' αυτήν. Στη δεύτερη γραμμή, με τη φράση:

WHENEVER ERROR ROLLBACK

απαιτούμε από το σύστημα, οποτεδήποτε συμβεί κάποιο σφάλμα να επιστρέψει στην προηγούμενη κατάσταση, ενώ με τη φράση:

WHENEVER συνθήκη λάθους ROLLBACK

απαιτούμε από το σύστημα, εφόσον ικανοποιηθεί η συνθήκη λάθους που έχουμε επιβάλλει, να επιστρέψει στην προηγούμενη κατάσταση.

Για παράδειγμα, ο λογιστής που καταχωρεί τους μισθούς, για να αποφύγει την περίπτωση λανθασμένης καταχώρισης, θα μπορούσε να δώσει για συνθήκη,

WHENEVER μισθός > 600.000 ROLLBACK

Έτσι, αν γράψει στη στήλη «μισθός» τον αριθμό 600.000 ή μεγαλύτερο, τότε αυτόματα ενεργοποιείται η ROLLBACK.

1.2.1.1 Παράδειγμα συναλλαγής

Ακολουθεί ένα παράδειγμα προγράμματος συναλλαγής (Greyet al,1981):

```
FUNDS_TRANSFER: PROCEDURE;
$BEGIN_TRANSACTION;
ON ERROR DO;                              /*στην περίπτωση λάθους*/
$RESTORE_TRANSACTION;                     /*αναιρείται όλη η εργασία*/
GET INPUT MESSAGE;                        /*απαιτείται εισαγωγή δεδομένων*/
PUT MESSAGE ('TRANSFER FAILED');          /*αναφορά σφάλματος της συναλλαγής*/
GO TO COMMIT;
END;
GET INPUT MESSAGE;                        /*εισαγωγή και ανάλυση δεδομένων*/
EXTRACT ACCOUNT_DEBIT, ACCOUNT_CREDIT,
```

AMOUNT FROM MESSAGE;

$UPDATE ACCOUNTS /*χρέωση*/

SET BALANCE =BALANCE - AMOUNT

WHERE ACCOUNTS_NUMBER = ACCOUNTS -DEBIT;

$UPDATE ACCOUNTS /*πίστωση*/

SET BALANCE = BALANCE + AMOUNT

WHERE ACCOUNTS = NUMBER - ACCOUNTS_CREDIT;

$INSERT INTO HISTORY (DATE, MESSAGE); /*αποθήκευση στοιχείων ελέγχου*/

PUT MESSAGE ('TRANSFER DONE'); /*αναφορά επιτυχίας της συναλλαγής*/

COMMIT: /*εκτέλεση ενημερώσεων*/

$COMMIT_TRANSACTION;

END; /*τέλος προγράμματος*/

1.3 Ιδιότητες συναλλαγών

Για να διασφαλιστεί η ακεραιότητα των δεδομένων, απαιτείται η βάση δεδομένων να διατηρεί τις παρακάτω ιδιότητες συναλλαγών. Οι ιδιότητες αυτές συνήθως ονομάζονται ιδιότητες **ACID** (Atomicity Consistency Isolation Durability). Το ακρωνύμιο παράγεται από το πρώτο γράμμα κάθε μιας από τις τέσσερις ιδιότητες στα αγγλικά (Gray,1981;Gray,1993).

- ***Ατομικότητα (Atomicity):*** Πρέπει να γίνουν όλες οι λεπτομέρειες της συναλλαγής σωστά στη βάση δεδομένων, ή καμία. Συγκεκριμένα είτε όλες οι πράξεις της συναλλαγής επιτυγχάνουν, είτε όλες αποτυγχάνουν.
- ***Συνέπεια (Consistency):*** Η εκτέλεση μιας συναλλαγής απομονωμένα (δηλαδή, χωρίς ταυτόχρονη εκτέλεση άλλων συναλλαγών) διατηρεί την συνέπεια της βάσης δεδομένων. Δηλαδή, στο τέλος της συναλλαγής, η βάση πρέπει να είναι σε συνεπή μορφή.
- ***Απομόνωση (Isolation):*** Ακόμα και αν εκτελούνται ταυτόχρονα πολλές συναλλαγές το σύστημα εγγυάται ότι, για κάθε ζευγάρι από συναλλαγές Ti και Tj, Θα φαίνεται στην Ti ότι η Tj τελείωσε την εκτέλεση πριν ξεκινήσει η Ti ή ότι η Tj ξεκίνησε την εκτέλεση αφού τελείωσε η Ti. Έτσι, κάθε συναλλαγή δεν ξέρει για τις άλλες συναλλαγές που εκτελούνται ταυτόχρονα στο σύστημα. Πιο αναλυτικά, κάθε συναλλαγή πρέπει να νομίζει ότι τρέχει μόνη της.
- ***Μονιμότητα (Durability):*** Αφού μια συναλλαγή ολοκληρωθεί με επιτυχία, παραμένουν οι αλλαγές που έχει κάνει στη βάση δεδομένων, ακόμα και αν το σύστημα έχει πρόβλημα.

1.3.1 Ατομικότητα

Η φράση "όλα ή τίποτα" περιγράφει με ακρίβεια την πρώτη ιδιότητα του ACID, την ατομικότητα. Όταν μια ενημέρωση λαμβάνει χώρα σε μια βάση δεδομένων, είτε όλα τα στοιχεία ή κανένα από αυτά της ενημέρωσης, γίνεται διαθέσιμη σε όλους εκτός από το χρήστη ή την εφαρμογή που εκτελεί την ενημέρωση. Αυτή η ενημερωμένη έκδοση για τη βάση δεδομένων ονομάζεται συναλλαγή και είτε εκτελείται ή ματαιώνεται. Αυτό σημαίνει ότι μόνο ένα μέρος της ενημέρωσης δεν μπορεί να αποθηκευθεί στη βάση δεδομένων, θα δημιουργηθούν προβλήματα είτε με το υλικό ή το λογισμικό. Τα χαρακτηριστικά που είναι απαραίτητο να λαμβάνονται υπόψη για την ατομικότητα είναι:

- μια συναλλαγή είναι ένα τμήμα μιας εργασίας- ή όλες οι ενέργειές της συναλλαγής ολοκληρώνονται ή τερματίζονται
- η ατομικότητα διατηρείται με την εμφάνιση αδιεξόδων (deadlocks)
- η ατομικότητα διατηρείται με την εμφάνιση σφαλμάτων του λογισμικού βάσεων δεδομένων
- η ατομικότητα διατηρείται με την εμφάνιση σφαλμάτων του λογισμικού εφαρμογών
- η ατομικότητα διατηρείται με την εμφάνιση σφαλμάτων της CPU
- η ατομικότητα διατηρείται με την εμφάνιση σφαλμάτων δίσκων
- η ατομικότητα μπορεί να απενεργοποιηθεί σε επίπεδο συστήματος
- η ατομικότητα μπορεί να απενεργοποιηθεί σε επίπεδο συνόδου

1.3.2 Συνέπεια

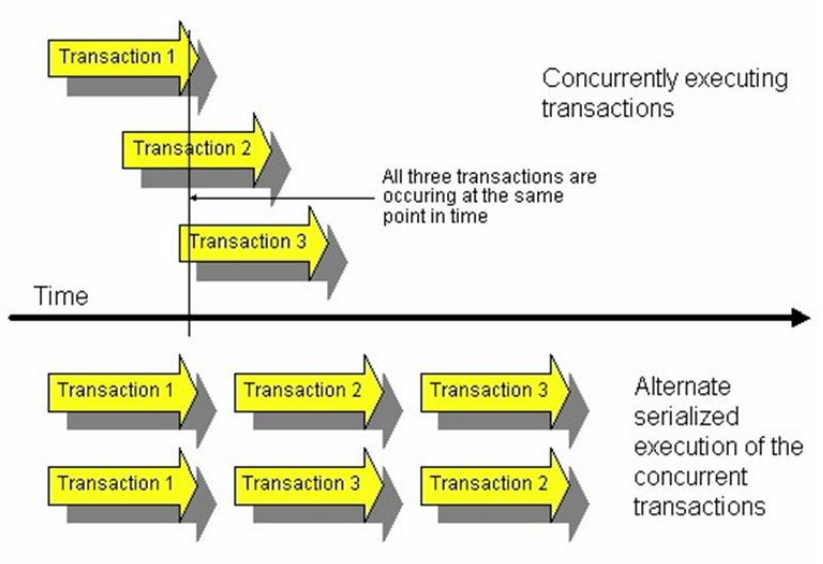
Συνέπεια είναι μία ιδιότητα του ACID, η οποία διασφαλίζει ότι οποιεσδήποτε αλλαγές σε τιμές σε μια περίπτωση, συνάδουν με τις αλλαγές σε άλλες τιμές

στην ίδια περίπτωση. Ένας περιορισμός της συνέπειας είναι ένα κατηγόρημα σε δεδομένα που χρησιμεύει ως προϋπόθεση, αποτέλεσμα, και κατάσταση μετασχηματισμού σε οποιαδήποτε συναλλαγή.

1.3.3 Απομόνωση

Η απομόνωση απαιτείται όταν πραγματοποιούνται ταυτόχρονες συναλλαγές. Οι ταυτόχρονες συναλλαγές είναι οι συναλλαγές που λαμβάνουν χώρα την ίδια στιγμή, όπως μια κοινή πρόσβαση πολλών χρηστών σε κοινόχρηστα αντικείμενα. Αυτή η κατάσταση απεικονίζεται στο επάνω μέρος του σχήματος ως δραστηριότητες που συμβαίνουν μέσα στην πάροδο του χρόνου. Η προστασία που προσφέρει ένα ΣΔΒΔ για να αποφευχθεί η σύγκρουση μεταξύ ταυτόχρονων συναλλαγών είναι μια έννοια που αναφέρεται ως απομόνωση.

Για παράδειγμα, αν δύο άνθρωποι ενημερώνουν το ίδιο στοιχείο καταλόγου, δεν είναι αποδεκτό για τις αλλαγές ενός ατόμου να «κατατροπώνονται» όταν το δεύτερο πρόσωπο αποθηκεύει ένα διαφορετικό σύνολο αλλαγών. Και οι δύο χρήστες θα πρέπει να είναι σε θέση να λειτουργούν μεμονωμένα, να εργαστούν σαν να είναι ο μόνος χρήστης. Κάθε σειρά αλλαγών πρέπει να απομονωθεί από εκείνες των άλλων χρηστών.



Εικόνα 3 - Εκτέλεση Συναλλαγών

Μια σημαντική έννοια για την κατανόηση της απομόνωσης μέσω συναλλαγών είναι η σειριοποιησιμότητα. Οι συναλλαγές είναι σειριοποιήσιμες όταν η επίδραση στη βάση δεδομένων είναι η ίδια αν οι συναλλαγές που εκτελούνται με αύξοντα αριθμό ή κατά διαπλεκόμενο τρόπο. Στο πάνω μέρος της εικόνας 3, η συναλλαγή Transactions 1 μέσω της συναλλαγής Transaction 3 εκτελείται ταυτόχρονα στην πάροδο του χρόνου. Το αποτέλεσμα στο ΣΔΒΔ είναι ότι οι συναλλαγές μπορούν να εκτελέσουν με αύξοντα αριθμό με βάση τις απαιτήσεις της συνοχής και της απομόνωσης. Στο κάτω μέρος της εικόνας 3, εμφανίζονται διάφοροι τρόποι με τους οποίους οι συναλλαγές αυτές μπορούν να εκτελεστούν. Είναι σημαντικό να σημειωθεί ότι η σειριοποιήσιμη εκτέλεση δεν συνεπάγεται τις πρώτες συναλλαγές που αυτομάτως θα τερματίσουν πριν από τις άλλες συναλλαγές με αύξοντα αριθμό.

Οι βαθμοί απομόνωσης είναι οι ακόλουθοι:

- **βαθμός 0** - μια συναλλαγή δεν επικαλύπτει ασταθείς αναγνώσεις (dirtyreads) δοσοληψιών με μεγαλύτερο βαθμό
- **βαθμός 1** - δεν παρουσιάζει απώλειες ενημέρωσης
- **βαθμός 2** - ούτε απώλειες ενημέρωσης, ούτε ασταθείς αναγνώσεις
- **βαθμός 3** - βαθμός 2 + επαναλήψιμες αναγνώσεις

1.3.4 Μονιμότητα

Η διατήρηση ενημερώσεων των εκτελούμενων συναλλαγών είναι συνήθως κρίσιμη. Αυτές οι ενημερώσεις δεν πρέπει ποτέ να χαθούν. Η ιδιότητα του ACID, μονιμότητα, πραγματεύεται την ανάγκη αυτή. Η μονιμότητα αναφέρεται στην ικανότητα του συστήματος να ανακτήσει εκτελούμενες ενημερώσεις συναλλαγών είτε εφόσον το σύστημα ή το μέσο αποθήκευσης αποτύχει.

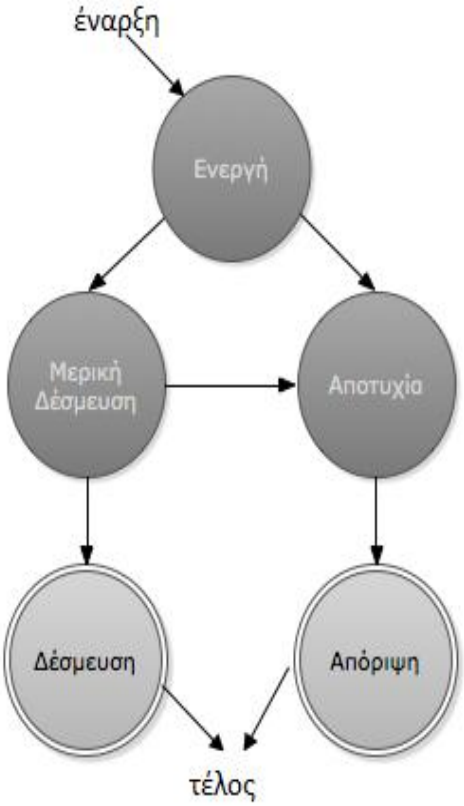
Τα χαρακτηριστικά που λαμβάνονται υπόψη για την μονιμότητα είναι:

- επαναφορά στην πιο πρόσφατη επιτυχημένη εκτέλεση μετά από μια αποτυχία του λογισμικού βάσεων δεδομένων
- επαναφορά στην πιο πρόσφατη επιτυχημένη εκτέλεση μετά από μια αποτυχία του λογισμικού εφαρμογής
- επαναφορά στην πιο πρόσφατη επιτυχημένη εκτέλεση μετά από μια αποτυχία της CPU
- επαναφορά στην πιο πρόσφατη επιτυχημένη συλλογή αντιγράφων ασφαλείας μετά από μια αποτυχία του δίσκου
- επαναφορά στην πιο πρόσφατη επιτυχημένη εκτέλεση μετά από μια αποτυχία δίσκου δεδομένων

1.4 Καταστάσεις συναλλαγών

Μια συναλλαγή είτε θα τερματιστεί κανονικά (commited) επιφέροντας μόνιμες αλλαγές στη ΒΔ είτε θα απορριφθεί (aborted) και τα δεδομένα θα επανέλθουν στην προηγούμενη κατάσταση (Bernstein et al,2009). Οι δυο αυτές καταστάσεις είναι οι μοναδικές καταστάσεις τερματισμού μιας συναλλαγής. Ωστόσο, πριν τον τερματισμό της μια συναλλαγή μπορεί να βρεθεί σε άλλη ενδιάμεση κατάσταση. Οι καταστάσεις στις οποίες μπορεί να υπεισέλθει μια συναλλαγή παρουσιάζονται στο σχήμα δεξιά και περιγράφονται στη συνέχεια.

- **Ενεργή Κατάσταση (active).** Η συναλλαγή εισέρχεται στην ενεργή κατάσταση κατά την αρχή της επεξεργασίας της και παραμένει σε αυτή ενόσω εκτελείται.
- **Κατάσταση Μερικής Δέσμευσης (partiallycommited).** Η συναλλαγή θεωρείται μερικώς ολοκληρωμένη όταν έχει ολοκληρωθεί και η τελευταία εντολή της συναλλαγής.
- **Κατάσταση Αποτυχίας (failed).** Η συναλλαγή αποτυγχάνει όταν το ΣΔΒΔ αντιληφθεί ότι δεν μπορεί να συνεχίσει η ομαλή επεξεργασία της.
- **Κατάσταση Απόρριψης (aborted).** Η συναλλαγή βρίσκεται στην κατάσταση αυτή όταν τα δεδομένα έχουν επανέλθει στην προηγούμενη σταθερή κατάσταση, πριν την αρχή της εκτέλεσης της συναλλαγής.
- **Κατάσταση Δέσμευσης (commited).** Η συναλλαγή έχει ολοκληρώσει την εκτέλεση της με επιτυχία.



Εικόνα 4 - Καταστάσεις Συναλλαγών

1.5 Χρονοπρογράμματα

1.5.1 Τι είναι

Ένα χρονοπρόγραμμα ενός συστήματος, στη γνωστική περιοχή των Βάσεων Δεδομένων και στην Επεξεργασία Συναλλαγών, είναι ένα αφηρημένο μοντέλο που περιγράφει την εκτέλεση συναλλαγών που τρέχουν στο σύστημα. Συχνά είναι μια λίστα από ενέργειες οι οποίες αναπαρίστανται με χρονική σειρά και εκτελούνται από ένα σύνολο συναλλαγών οι οποίες με την σειρά τους εκτελούνται όλες μαζί στο σύστημα. Παραδείγματα αυτών των ενεργειών είναι τα read, write, abort , commit, αίτηση κλειδώματος, κλείδωμα κοκ. Δεν λαμβάνουν μέρος όλα τα είδη ενεργειών των συναλλαγών σε ένα χρονοπρόγραμμα και μόνο προεπιλεγμένα είδη ενεργειών συμπεριλαμβάνονται, που χρειάζονται για να περιγράψουν συγκεκριμένα φαινόμενα. Τα χρονοπρογράμματα και οι ιδιότητες χρονοπρογραμμάτων είναι απαραίτητα στοιχεία για τον έλεγχο ταυτοχρονισμού σε μια Βάση Δεδομένων.

Ένα παράδειγμα χρονοπρογράμματος παρουσιάζεται στην εικόνα 5. Σε αυτό το παράδειγμα ο οριζόντιος άξονας αναπαριστά τις διαφορετικές συναλλαγές στο χρονοπρόγραμμα D ενώ ο κάθετος άξονας αναπαριστά την χρονική σειρά των εργασιών. Το χρονοπρόγραμμα D αποτελείται από τρεις συναλλαγές T1, T2, T3.Το χρονοπρόγραμμα περιγράφει τις ενέργειες των συναλλαγών όπως τις βλέπουμε σε ένα Σύστημα Διαχείρισης Βάσης Δεδομένων. Αρχικά η T1 διαβάζει και γράφει το αντικείμενο X και μετά κάνει εκτελεί την εντολή commit (η συναλλαγή τερματίζει επιτυχώς). Η T2 διαβάζει και γράφει το αντικείμενο Y και μετά εκτελεί την εντολή commit και η T3 διαβάζει και γράφει το αντικείμενο Z και μετά κάνει commit. Αυτό είναι ένα παράδειγμα σειρακού χρονοπρογράμματος επειδή οι ενέργειες των τριών συναλλαγών δεν περιπλέκονται μεταξύ τους.

$$D = \begin{bmatrix} T1 & T2 & T3 \\ R(X) & & \\ W(X) & & \\ Com. & & \\ & R(Y) & \\ & W(Y) & \\ & Com. & \\ & & R(Z) \\ & & W(Z) \\ & & Com. \end{bmatrix}$$

Εικόνα 5- Παράδειγμα χρονοπρογράμματος

Το χρονοπρόγραμμα D αναπαρίσταται από ένα πίνακα και αυτό γίνεται για διευκόλυνση του χρήστη ώστε να βλέπει κάθε συναλλαγή. Ένας πιο κοινός τρόπος περιγραφής του χρονοπρογράμματος είναι με την χρήση λίστας όπως φαίνεται παρακάτω:

D = R1(X) W1(X) Com1 R2(Y) W2(Y) Com2 R3(Z) W3(Z) Com3

Συνήθως για την αιτιολόγηση του ελέγχου ταυτοχρονισμού στις ΒΔ, μία ενέργεια χαρακτηρίζεται σαν ατομική όταν προκύπτει σε μια χρονική στιγμή (χωρίς διάρκεια). Όταν αυτό δεν είναι ικανοποιητικό, τότε προσδιορίζονται χρονικά σημεία αρχής και τέλους και πιθανότατα και άλλα σημεία γεγονότων (πιο σπάνια). Οι ενέργειες που εκτελούνται έχουν πάντα συγκεκριμένο χρόνο εκτέλεσης (πχ ακριβής χρόνος μέχρι την ολοκλήρωση τους), όμως για την αιτιολόγηση του ελέγχου ταυτοχρονισμού έχει σημασία μόνο η χρονική προτεραιότητα. Επιπρόσθετα σε πολλές περιπτώσεις οι πριν/μετά σχέσεις μεταξύ δυο ενεργειών δεν έχουν σημασία και δεν ορίζονται, αφού ορίζονται για άλλους συνδυασμούς ενεργειών (Weikum et al, 2001).

Γενικά οι ενέργειες των συναλλαγών σε ένα χρονοπρόγραμμα μπορούν να περιπλέκουν (εκτελούνται ταυτόχρονα), καθώς η χρονική σειρά των λειτουργιών σε κάθε συναλλαγή δεν αλλάζει όπως προϋποθέτει το

πρόγραμμα της συναλλαγής. Αφού δεν έχει μόνο σημασία η χρονική σειρά όλων των συναλλαγών, πρέπει να οριστούν κιόλας, έτσι ορίζεται ένα χρονοπρόγραμμα. Γενικά επιλέγεται μια μερική διαμέριση μεταξύ των ενεργειών αντί μιας ολικής όπου η διαμέριση για κάθε συνδυασμό διευκρινίζεται όπως σε μια λίστα ενεργειών. Επίσης σε γενικές περιπτώσεις κάθε συναλλαγή μπορεί να αποτελείται από πολλές διεργασίες και μπορεί να αναπαρίσταται από μερική διαμέριση διεργασιών παρά από μια ολική διαμέριση. Έτσι γενικά ένα χρονοπρόγραμμα είναι η μερική διαμέριση ενεργειών που περιέχουν (ενσωματώνουν) τις μερικές διαμερίσεις όλων των συναλλαγών τους.

Η χρονική σειρά μεταξύ δυο ενεργειών μπορεί να αναπαρασταθεί από ένα ζεύγος αυτών των ενεργειών (π.χ. η ύπαρξη ενός ζεύγους (OP1,OP2) σημαίνει ότι η OP1 είναι πάντα πριν από την ενέργεια OP2), και ένα χρονοπρόγραμμα είναι ένα σύνολο τέτοιων ορισμένων ζευγών. Ένα τέτοιο σύνολο, δηλαδή ένα χρονοπρόγραμμα μπορεί να αναπαρασταθεί από ένα μη κυκλικό κατευθυνόμενο γράφο με τις ενέργειες σαν κόμβους και την χρονική σειρά σαν ακμές (δεν επιτρέπονται κύκλοι αφού κύκλος σημαίνει ότι μια λειτουργία μπορεί να γίνει και πριν αλλά και μετά από μια άλλη, που αντιτίθεται με την αντίληψη μα για τον χρόνο). Σε πολλές περιπτώσεις μια γραφική αναπαράσταση τέτοιων γράφων χρησιμοποιείται για να αναπαραστήσει ένα χρονοπρόγραμμα.

1.5.2 Σειριακό χρονοπρόγραμμα

Οι συναλλαγές εκτελούνται μη περιπλεγμένα (εικόνα 6), δηλαδή σειριακό χρονοπρόγραμμα είναι ένα χρονοπρόγραμμα στο οποίο καμία συναλλαγή δεν αρχίζει πριν τελειώσει μια συναλλαγή η οποία ήδη τρέχει (Bernstein et al,1987).

$$E = \begin{bmatrix} T1 & T2 & T3 \\ R(X) & & \\ & R(Y) & \\ & & R(Z) \\ W(X) & & \\ & W(Y) & \\ & & W(Z) \\ Com. & Com. & Com. \end{bmatrix}$$

Εικόνα 6-Παράδειγμα σειριακού χρονοπρογράμματος

1.5.3 Σειριοποιήσιμο χρονοπρόγραμμα

Ένα χρονοπρόγραμμα που είναι ισοδύναμο ως προς το αποτέλεσμα με ένα σειριακό χρονοπρόγραμμα όχι με την ιδιότητα της σειριοποιησιμότητας. Στο χρονοπρόγραμμα Ε η σειρά με την οποία εκτελούνται οι πράξεις δεν είναι ίδια όπως το D αλλά έχει τα ίδια αποτελέσματα με το D.

1.6 Συγκρούσεις

Δύο οι περισσότερες ενέργειες καταλήγουν σε σύγκρουση αν συμβαίνουν τα εξής:

1. Οι ενέργειες ανήκουν σε διαφορετικές συναλλαγές.
2. Τουλάχιστον μια από τις δύο ενέργειες είναι write.
3. Οι ενέργειες αυτές προσπελαύνουν το ίδιο αντικείμενο.

Το ακόλουθο σύνολο ενεργειών οδηγεί σε σύγκρουση:

- T1:R(X), T2:W(X), T3:W(X)

Ενώ το ακόλουθο σύνολο ενεργειών δεν οδηγεί σε σύγκρουση:

- T1:R(X), T2:R(X), T3:R(X)
- T1:R(X), T2:W(Y), T3:R(X)

Σχόλιο [Φ1]: Μπορείς να το εξηγήσεις με λίγα λόγια

1.6.1 Ισοδυναμία συγκρούσεων

Τα χρονοπρογράμματα S1, S2 μπορούμε να πούμε ότι είναι ισοδύναμα συγκρούσεων αν ισχύουν τα ακόλουθα:

1. Και τα δυο χρονοπρογράμματα περιπλέκουν το ίδιο σύνολο από συναλλαγές
2. Η σειρά συγκρουόμενων ενεργειών στα χρονοπρογράμματα S1 και S2 είναι ίδια.

1.6.2 Σειριοποιησιμότητα συγκρούσεων

Ένα χρονοπρόγραμμα είναι σειριοποιήσιμο συγκρούσεων όταν το χρονοπρόγραμμα είναι ισοδύναμο συγκρούσεων με ένα ή περισσότερα σειριακά χρονοπρογράμματα.

Μια άλλη εκδοχή για την σειριοποιησιμότητα συγκρούσεων είναι ότι ένα χρονοπρόγραμμα είναι σειριοποιήσιμο συγκρούσεων αν και μόνο αν υπάρχει ένας μη κυκλικός γράφος σειριοποιησιμότητας για το χρονοπρόγραμμα. Το οποίο είναι ισοδύναμο συγκρούσεων με το σειριακό χρονοπρόγραμμα <T1,T2> αλλά όχι με το <T2,T1> (Εικόνα 7).

$$G = \begin{bmatrix} T1 & T2 \\ R(A) & \\ & R(A) \\ W(B) & \\ Com. & \\ & W(A) \\ & Com. \end{bmatrix}$$

Εικόνα 7 - Παράδειγμα χρονοπρογράμματος με συγκρούσεις

1.6.3 Ισοδυναμία όψεως

Δυο χρονοπρογράμματα S1, S2 είναι ισοδύναμα όψεως όταν ισχύουν οι ακόλουθες συνθήκες:

- Τα S1 και S2 περιέχουν το ίδιο σύνολο δοσοληψιών και τις ίδιες πράξεις των συγκεκριμένων δοσοληψιών
- Για κάθε στοιχείο X, αν η Ti διαβάζει την αρχική του τιμή στο S1, πρέπει να διαβάζει επίσης την αρχική τιμή στο S2
- Για κάθε πράξη Ri(X) στο S1 μετά από μια πράξη Wj(X) πρέπει να ισχύει η ίδια συνθήκη και στο S2.

- Αν η πράξη Wi(Y) στο S1 είναι η τελευταία πράξη που τροποποιεί το Y τότε πρέπει να ισχύει η ίδια συνθήκη και στο S2.

1.6.4 Σειριοποιησιμότητα όψεως

Ένα χρονοπρόγραμμα είναι σειριοποιήσιμο όψεως αν είναι ισοδύναμο όψεως με μερικά σειριακά χρονοπρογράμματα. Εξ' ορισμού όλα τα σειριοποιήσιμα συγκρούσεως είναι σειριοποιήσιμα όψεως.

$$G = \begin{bmatrix} T1 & T2 \\ R(A) & \\ & R(A) \\ W(B) & \end{bmatrix}$$

Εικόνα 8 - Παράδειγμα χρονοπρογράμματος με σειριοποιησιμότητα συγκρούσεων

Το παραπάνω παράδειγμα (το οποίο είναι το ίδιο που παρουσιάστηκε στη σειριοποιησιμότητα συγκρούσεων) είναι τόσο σειριοποιήσιμο συγκρούσεων όσο και σειριοποιήσιμο όψεως. Υπάρχουν βέβαια χρονοπρογράμματα τα οποία είναι σειριοποιήσιμα όψεως αλλά όχι σειριοποιήσιμα συγκρούσεων, αυτά τα χρονοπρογράμματα συνήθως έχουν μια συναλλαγή η οποία γράφει ένα αντικείμενο χωρίς να το διαβάζει (blind write):

$$H = \begin{bmatrix} T1 & T2 & T3 \\ R(A) & & \\ & W(A) & \\ & Com. & \\ W(A) & & \\ Com. & & \\ & & W(A) \\ & & Com. \end{bmatrix}$$

Εικόνα 9 - Παράδειγμα χρονοπρογράμματος με σειριοποιησιμότητα συγκρούσεων

Το παραπάνω παράδειγμα δεν είναι σειριοποιήσιμο συγκρούσεων αλλά είναι σειριοποιήσιμο όψεως μια και είναι σειριακό χρονοπρόγραμμα ισοδύναμο όψεως <T1, T2, T3>.

1.6.5 Ανακτησιμότητα

Οι συναλλαγές κάνουν commit μόνο αφού όλες οι συναλλαγές οι οποίες έχουν αλλάξει έχουν διαβάσει το commit.

$$F = \begin{bmatrix} T1 & T2 \\ R(A) & \\ W(A) & \\ & R(A) \\ & W(A) \\ Com. & \\ & Com. \end{bmatrix} F2 = \begin{bmatrix} T1 & T2 \\ R(A) & \\ W(A) & \\ & R(A) \\ & W(A) \\ Abort & \\ & Abort \end{bmatrix}$$

Εικόνα 10 - Παράδειγμα ανακτησιμότητας συγκρούσεων

Αυτά τα χρονοπρογράμματα είναι ανακτήσιμα. Το F είναι ανακτήσιμο επειδή η T1 κάνει commit πριν την T2 και αυτό κάνει το διάβασμα ορθό. Έπειτα η T2 κάνει commit. Σε περίπτωση που η T1 κάνει abort τότε και η T2 κάνει abort επειδή το αντικείμενο που διαβάζει από το A δεν είναι σωστό. Και στις δυο περιπτώσεις υπάρχει συνέπεια.

1.6.5.1 Μη – ανακτήσιμο

Αν μια συναλλαγή T1 κάνει abort και η συναλλαγή T2 που βασίζεται πάνω στην T1 κάνει commit τότε έχουμε ένα μη ανακτήσιμο χρονοπρόγραμμα.

$$G = \begin{bmatrix} T1 & T2 \\ R(A) & \\ W(A) & \\ & R(A) \\ & W(A) \\ & Com. \\ Abort & \\ & \end{bmatrix}$$

Εικόνα 11 - Παράδειγμα μη ανακτήσιμου προγράμματος

Σε αυτό το παράδειγμα το G χρονοπρόγραμμα είναι μη-ανακτήσιμο επειδή η T2 διάβασε την T1 η οποία δεν έγινε commit και έτσι έχουμε μια εσφαλμένη T2 αφού μετά μαθαίνουμε ότι η T1 έχει γίνει abort.

1.6.5.2 Αποφυγή διαδοχικών abort

Ένα abort σε μια συναλλαγή μπορεί να οδηγήσει σε πολλά περισσότερα. Μια στρατηγική για να αποφευχθεί αυτό είναι να μην επιτρέπεται από μια συναλλαγή να διαβάζει συναλλαγές οι οποίες δεν έχουν γίνει commit από το ίδιο χρονοπρόγραμμα. Το ακόλουθο παράδειγμα είναι το ίδιο με το παραπάνω όταν αναφέραμε την ανακτησιμότητα:

$$F = \begin{bmatrix} T1 & T2 \\ R(A) & \\ W(A) & \\ & R(A) \\ & W(A) \\ Com. & \\ & Com. \end{bmatrix} \quad F2 = \begin{bmatrix} T1 & T2 \\ R(A) & \\ W(A) & \\ & R(A) \\ & W(A) \\ Abort & \\ & Abort \end{bmatrix}$$

Εικόνα 12 - Παράδειγμα χρονοπρογραμμάτων abort

Σε αυτό το παράδειγμα παρόλο που η F2 είναι ανακτήσιμη δεν μπορεί να αποφύγει τα διαδοχικά aborts. Μπορούμε να δούμε αν κάνει abort η T1 τότε και η T2 θα πρέπει να γίνει abort για να διατηρηθεί η ορθότητα.

Το παρακάτω είναι ένα ανακτήσιμο χρονοπρόγραμμα το οποίο αποφεύγει διαδοχικά abort. Όμως χάνεται το γράψιμο της T1. Η τεχνική αποφυγής διαδοχικών abort αλλά όχι απαραίτητη για χρονοπρογράμματα έτσι ώστε να είναι ανακτήσιμα.

$$F3 = \begin{bmatrix} T1 & T2 \\ & R(A) \\ R(A) & \\ W(A) & \\ & W(A) \\ Abort & \\ & Commit \end{bmatrix}$$

Εικόνα 13 - Παράδειγμα ανακτήσιμου προγράμματος με αποφυγή abort

1.6.6 Αυστηρότητα

Ένα χρονοπρόγραμμα είναι αυστηρό, έχει την ιδιότητα strictness, αν για οποιαδήποτε από τις δυο συναλλαγές T1, T2 αν μια ενέργεια διαβάσματος της T1 οδηγήσει σε σύγκρουση με την T2 (είτε με διάβασμα είτε με γράψιμο), τότε το commit της T1 επίσης οδηγεί σε σύγκρουση με την ενέργεια T2.

Οποιοδήποτε αυστηρό χρονοπρόγραμμα είναι μη – διαδοχικών abort, αλλά όχι το αντίθετο. Η ιδιότητα αυτή είναι αποτελεσματική στην ανακτησιμότητα και αποτρέπει αποτυχίες στην βάση δεδομένων.

2 ο ΚΕΦΑΛΑΙΟ

ΤΑΥΤΟΧΡΟΝΗ ΕΚΤΕΛΕΣΗ ΠΟΛΛΩΝ ΣΥΝΑΛΛΑΓΩΝ

2.1 Εισαγωγή

Η εισαγωγή της έννοιας του χρονοπρογράμματος, που προηγήθηκε, θα μας βοηθήσει να περιγράψουμε τις εναλλασσόμενες εκτελέσεις συναλλαγών. Το ΣΔΒΔ επιδιώκει κάτι τέτοιο, ώστε να βελτιώσει τις επιδόσεις του συστήματος, δηλαδή να αυξήσει την απόδοση επεξεργασίας ή το χρόνο απόκρισης για σύντομες συναλλαγές, όμως δεν είναι όλες οι εναλλασσόμενες εκτελέσεις επιτρεπτές. Σε αυτό το κεφάλαιο, θα θεωρήσουμε τις εναλλασσόμενες εκτελέσεις, ή χρονοπρογράμματα, που είναι επιτρεπτές από ένα ΣΔΒΔ.

Για να δοθεί στους χρήστες ένας απλός τρόπος για να αντιληφθούν το αποτέλεσμα που επιφέρει η εκτέλεση των προγραμμάτων τους, η εναλλαγή των συναλλαγών ή διεργασιών γίνεται πολύ προσεκτικά, ώστε να εξασφαλιστεί το γεγονός ότι το αποτέλεσμα της ταυτόχρονης εκτέλεσης των συναλλαγών θα είναι ισοδύναμο (ως προς τις μεταβολές που προκαλούνται στη βάση δεδομένων) με κάποια σειριακή εκτέλεση των ίδιων συναλλαγών, δηλαδή, μια εκτέλεση όπου οι ενέργειες της κάθε συναλλαγής ολοκληρώνονται, πριν αρχίσει να εκτελείται η επόμενη συναλλαγή..

Ο σωστός συντονισμός των συναλλαγών είναι ευθύνη του ΣΔΒΔ, το οποίο μέσω του μηχανισμών ελέγχου ταυτοχρονισμού (concurrency control), εγγυάται την ομαλή εκτέλεση των συναλλαγών. Ο έλεγχος ταυτοχρονισμού μπορεί να επιτευχθεί με αλγορίθμους χρονοδρομολόγησης των διεργασιών. Θα εστιάσουμε, συγκεκριμένα στο μηχανισμό κλειδώματος των διεργασιών, ο οποίος θα περιγραφεί σε επόμενη ενότητα. Σκοπός του κεφαλαίου αυτού είναι η ενασχόληση με την αναγκαιότητα της ταυτόχρονης εκτέλεσης, αλλά και με τα προβλήματα που αυτή συνεπάγεται.

2.2 Πλεονεκτήματα από την ταυτόχρονη εκτέλεση συναλλαγών

Οι λόγοι που οδήγησαν ερευνητές και κατασκευαστές στη μελέτη της ταυτόχρονης εκτέλεσης συναλλαγών είναι, όπως προαναφέρθηκε, να βελτιωθούν οι επιδόσεις του συστήματος. Αναλυτικότερα:

- Μια συναλλαγή αποτελείται από πολλά διαφορετικά τμήματα εκ των οποίων, άλλα απαιτούν περισσότερο χρόνο από την κεντρική μονάδα επεξεργασίας (CPU) και άλλα απαιτούν περισσότερες προσπελάσεις στο δίσκο (I/O). Ο δίσκος και η CPU του συστήματος μπορούν να λειτουργούν παράλληλα, εξυπηρετώντας, έτσι, διαφορετικά τμήματα της ίδιας συναλλαγής ή διαφορετικών συναλλαγών. Το χρονικό διάστημα κατά το οποίο ο δίσκος του συστήματος είναι απασχολημένος με την ανάγνωση ή αποθήκευση δεδομένων εκ μέρους μίας συναλλαγής, η CPU μπορεί να χρησιμοποιηθεί για την εκτέλεση υπολογισμών για λογαριασμό μίας άλλης συναλλαγής. Το αποτέλεσμα είναι ότι αυξάνεται ο αριθμός των συναλλαγών που ολοκληρώνονται στη μονάδα του χρόνου (throughput).
- Ένα σύνολο συναλλαγών μπορεί να περιλαμβάνει συναλλαγές που έχουν μικρές απαιτήσεις και μπορούν να ολοκληρωθούν γρήγορα και συναλλαγές με περισσότερες απαιτήσεις, η εκτέλεση των οποίων

μπορεί να απαιτεί μεγάλα χρονικά διαστήματα. Χρησιμοποιώντας σειριακή εκτέλεση συναλλαγών μία μικρή συναλλαγή, ίσως, χρειασθεί να περιμένει πολύ χρόνο για την ολοκλήρωση μίας μεγαλύτερης. Η ταυτόχρονη εκτέλεση των συναλλαγών συμβάλλει στη μείωση των καθυστερήσεων (waiting time) και του μέσου χρόνου εκτέλεσης των συναλλαγών (mean response time).

2.3 Προβλήματα από την εναλλασσόμενη εκτέλεση συναλλαγών

Η ταυτόχρονη εκτέλεση πολλών συναλλαγών, ωστόσο, ενδέχεται να επιφέρει προβλήματα στην ακεραιότητα και συνέπεια των δεδομένων της ΒΔ. Αν όλες οι συναλλαγές πραγματοποιούν μόνο λειτουργίες ανάγνωσης, τότε δεν υπάρχει κίνδυνος για τα δεδομένα. Όμως, ενδέχεται κάποιες από τις συναλλαγές να πραγματοποιούν ενημερώσεις σε δεδομένα που προσπελαύνονται από άλλες. Επομένως, πρέπει να υπάρχουν αξιόπιστοι μηχανισμοί ελέγχου της ακεραιότητας των δεδομένων καθώς πολλές συναλλαγές μπορεί να βρίσκονται σε εξέλιξη, προσπελαύνοντας (πιθανότατα) κοινά δεδομένα. Αν επιτραπεί σε δύο συναλλαγές να ενημερώσουν τα ίδια δεδομένα, τότε υπάρχει μεγάλη πιθανότητα αλλοίωσης των δεδομένων (Μανωλόπουλος, 2003).

Παρακάτω περιγράφονται τρεις περιπτώσεις όπου ο συντονισμός των συναλλαγών είναι απαραίτητος και μπορεί να επιλύσει σημαντικά προβλήματα που εμφανίζονται από την ταυτόχρονη εκτέλεση των συναλλαγών.

2.3.1 1η περίπτωση - Το πρόβλημα της χαμένης ενημέρωσης (lost update problem)

Στην πρώτη περίπτωση, εξετάζουμε το πρόβλημα της χαμένης ενημέρωσης (lost update problem).Το πρόβλημα αυτό μπορεί να εμφανιστεί όταν δύο

συναλλαγές πραγματοποιούν ανάγνωση και αποθήκευση στα ίδια δεδομένα. Έστω, οι δυο συναλλαγές που απεικονίζονται στο Σχήμα (α) της εικόνας 14, όπου παρουσιάζεται η αλληλουχία εκτέλεσης των εντολών των δυο συναλλαγών. Η συναλλαγή Σ2 διαβάζει την τιμή του Χ πριν να καταχωρηθεί η νέα τιμή από τη Σι, και αποθηκεύει τη νέα τιμή του Χ στη ΒΔ μετά από τη Σι. Το αποτέλεσμα της εκτέλεσης αυτής είναι ότι η ενημέρωση του Χ από τη Σι έχει χαθεί.

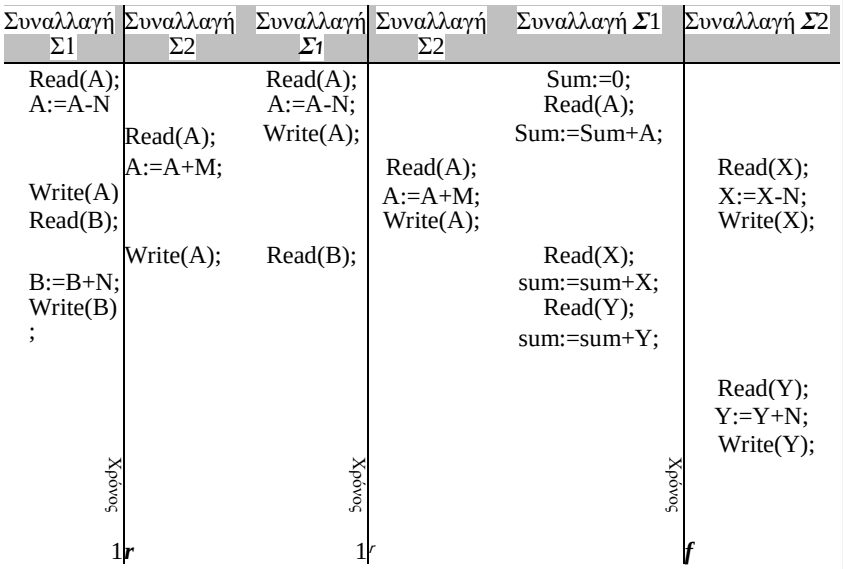
2.3.2 2η περίπτωση - Το πρόβλημα της λανθασμένης ανάγνωσης (dirty read)

Το δεύτερο πρόβλημα που εξετάζουμε είναι το πρόβλημα της λανθασμένης ανάγνωσης (dirty read), το οποίο απεικονίζεται στο Σχήμα (β) της εικόνας 14. Το πρόβλημα αυτό εμφανίζεται όταν μία συναλλαγή πραγματοποιεί αλλαγές στα δεδομένα και για κάποιο λόγο αποτυγχάνει. Πριν την επαναφορά των δεδομένων στην αρχική κατάσταση κάποια άλλη συναλλαγή διαβάζει τα δεδομένα με αποτέλεσμα να πραγματοποιείται μία λανθασμένη ανάγνωση. Στο παράδειγμα του σχήματος, η συναλλαγή Σι αλλάζει την τιμή του Χ και έστω, ότι αμέσως μετά αποτυγχάνει. Αμέσως μετά, η συναλλαγή Σ2 διαβάζει την τιμή του Χ, προσθέτει το Μ και στη συνέχεια αποθηκεύει τη νέα τιμή του Χ. Η τιμή του Χ που διαβάζεται από τη Σ2 είναι λανθασμένη, διότι η Σι έχει αποτύχει και επομένως, η τιμή του Χ έπρεπε να έχει αποκατασταθεί με την προηγουμένη τιμή (πριν την αλλαγή της Σι).

2.3.3 3η περίπτωση - Το πρόβλημα της παραγωγής συγκεντρωτικών στοιχείων

Το τρίτο πρόβλημα που εξετάζουμε αφορά στην παραγωγή συγκεντρωτικών στοιχείων (π.χ. άθροισμα, μέσο όρο) από τα δεδομένα της ΒΔ. Πιο συγκεκριμένα εξετάζουμε το πρόβλημα της λανθασμένης άθροισης (incorrect summary) το οποίο απεικονίζεται στο Σχήμα (γ) της εικόνας 14. Η

συναλλαγή Σι διαβάζει κάποια δεδομένα και παράγει το άθροισμα των δεδομένων αυτών. Ωστόσο, κάποια δεδομένα ενημερώνονται από τη Σ2 πριν να διαβασθούν από τη Σ1 (όπως το Χ), ενώ κάποια άλλα δεδομένα ενημερώνονται από τη Σ2 μετά την ανάγνωση τους από τη Σ1 (όπως το Υ). Το τελικό αποτέλεσμα της άθροισης είναι λανθασμένο.



Εικόνα 14 - Προβλήματα που προκύπτουν από την ταυτόχρονη εκτέλεση συναλλαγών

2.4 Συγκρούσεις προς αποφυγή

Στην ενότητα αυτή εξετάζονται τρεις τρόπους με τους οποίους ένα χρονοπρόγραμμα που περιλαμβάνει δύο ολοκληρωμένες συναλλαγές, που διατηρούν τη συνέπεια της βάσης, μπορεί να εκτελεστεί σε μια συνεπή βάση και να την καταστήσει ασυνεπή.

Όπως έχει προαναφερθεί, δύο ενέργειες πάνω σε ένα αντικείμενο της βάσης συγκρούονται, αν έστω, μια από αυτές, είναι ενέργεια εγγραφής (write).

Οι τρεις ανώμαλες περιπτώσεις μπορούν να περιγραφούν με βάση τις συγκρουόμενες ενέργειες δύο συναλλαγών Σ1 και Σ2: σε μια σύγκρουση write-read (WR) η Σ2 διαβάζει ένα αντικείμενο της βάσης το οποίο έχει προηγουμένως γραφεί από την Σ1. Παρόμοια, ορίζουμε και τις συγκρούσεις read-write (RW) και write-write (WW).

2.4.1 Ανάγνωση μη ολοκληρωμένων δεδομένων (Συγκρούσεις WR)

Η πρώτη περίπτωση ανωμαλίας είναι όταν μια συναλλαγή Σ2 μπορεί να διαβάσει ένα αντικείμενο της βάσης Α που έχει τροποποιηθεί από μια άλλη συναλλαγή Σ1, η οποία, όμως, δεν έχει ακόμα ολοκληρωθεί. Αυτό ονομάζεται **ασυνεπής ανάγνωση**.

Ένα απλό παράδειγμα δείχνει πως ένα τέτοιο χρονοπρόγραμμα μπορεί να οδηγήσει τη βάση σε ασυνεπή κατάσταση. Υποθέτουμε δύο συναλλαγές Σ1 και Σ2, κάθε μια από τις οποίες όταν εκτελεστεί από μόνη της διατηρεί τη συνέπεια της βάσης: η Σ1 μεταφέρει €100 από το Α στο Β, και η Σ2 αυξάνει και το Α και το Β κατά 6% (π.χ., καταθέτει τους ετήσιους τόκους στους δύο λογαριασμούς). Ας θεωρηθεί ότι οι ενέργειές τους εκτελούνται εναλλασσόμενα, έτσι ώστε (1) το πρόγραμμα μεταφοράς Σ1 να χρεώνει €100 τον Α, έπειτα (2) το πρόγραμμα κατάθεσης τόκων Σ2 να διαβάσει τις τρέχουσες τιμές των λογαριασμών Α και Β και να προσθέτει τόκους 6% στον καθένα, και τέλος (3) το πρόγραμμα μεταφοράς Σι να πιστώνει €100 στο λογαριασμό Β.

Το αντίστοιχο χρονοπρόγραμμα, που είναι ο τρόπος με τον οποίο το ΣΔΒΔ αντιλαμβάνεται τη σειρά των γεγονότων που συμβαίνουν, περιγράφεται σχηματικά στην Εικόνα 15.

Το αποτέλεσμα αυτού του χρονοπρογράμματος είναι διαφορετικό από όλα τα αποτελέσματα που θα μπορούσαμε να έχουμε εκτελώντας μια από τις δύο συναλλαγές και μετά την άλλη. Το πρόβλημα έγκειται στο γεγονός πως η

τιμή του Α που έγραψε η Σ1 διαβάστηκε από την Σ2, πριν η Σ1 ολοκληρώσει τις μεταβολές που είχε να επιτελέσει.

Σ1	Σ2
R(A)	
W(A)	
	R(A)
	W(A)
	R(B)
	W(B)
	Commit
R(B)	
W(B)	
Commit	

Εικόνα 15 - Χρονοπρόγραμμα με συγκρούσεις WR

Το γενικό πρόβλημα που περιγράφεται εδώ είναι ότι η Σι μπορεί να γράψει κάποια τιμή στο Α που κάνει τη βάση ασυνεπή. Εφόσον η Σι, πριν ολοκληρωθεί, επανεγγράψει αυτή την τιμή με μια «σωστή» τιμή για το Α, δεν υπάρχει πρόβλημα αν η Σι και η Σ2 εκτελεστούν με κάποια σειρά, επειδή η Σ2 δε θα δει την (προσωρινή) ασυνέπεια. Αντίθετα, μια εναλλασσόμενη εκτέλεση μπορεί να φανερώσει αυτή την ασυνέπεια και να οδηγήσει σε μια ασυνεπή τελική κατάσταση της βάσης.

Ας σημειωθεί, πως, παρόλο που μια συναλλαγή πρέπει μετά την ολοκλήρωση της να αφήνει τη βάση σε μια συνεπή κατάσταση, δεν είναι υποχρεωτικό να διατηρεί τη βάση συνεπή κατά τη διάρκεια της εκτέλεσης της. Κάτι τέτοιο θα ήταν πολύ περιοριστικό: για να μεταφέρει χρήματα από ένα λογαριασμό σε έναν άλλο, μια συναλλαγή πρέπει να χρεώσει τον ένα λογαριασμό, προσωρινά, αφήνοντας τη βάση ασυνεπή, και μετά να πιστώσει τον άλλο λογαριασμό, επαναφέροντας ξανά τη συνέπεια στη βάση.

2.4.2 Μη επαναλαμβανόμενες αναγνώσεις (Συγκρούσεις RW)

Η δεύτερη περίπτωση, στην οποία μπορεί να εμφανιστεί ανώμαλη συμπεριφορά, είναι όταν μια συναλλαγή Σ2 θα μπορούσε να αλλάξει την τιμή

ενός αντικειμένου Α που έχει διαβαστεί από μια συναλλαγή Σι, ενώ η Σι είναι ακόμα ενεργή. Κάτι τέτοιο δημιουργεί δύο προβλήματα.

Πρώτον, αν η Σ1 προσπαθήσει να ξαναδιαβάσει την τιμή του Α, θα πάρει μια διαφορετική τιμή, παρόλο που εντωμεταξύ η ίδια δεν έχει αλλάξει το Α. Η κατάσταση αυτή, που δεν μπορεί να συμβεί σε μια σειριακή εκτέλεση των δύο συναλλαγών, ονομάζεται μη επαναλαμβανόμενη ανάγνωση.

Δεύτερον, ας θεωρηθεί πως και η Σ1 και η Σ2 διαβάζουν την ίδια τιμή του Α, έστω 5, και έπειτα η Σ1 που θέλει να αυξήσει την τιμή του Α κατά 1 την αλλάζει σε 6, και η Σ2 που θέλει να μειώσει την τιμή του Α κατά 1, μειώνει την τιμή που είχε διαβάσει (δηλαδή 5) αλλάζοντας το Α σε 4. Η σειριακή εκτέλεση αυτών των συναλλαγών με οποιαδήποτε σειρά θα έδινε στο Α την τελική τιμή 5. Συνεπώς η εναλλασσόμενη εκτέλεση οδηγεί σε μια ασυνεπή κατάσταση.

Στην περίπτωση αυτή, η ουσία του προβλήματος έγκειται στο ότι ενώ η αλλαγή της Σ2 δεν διαβάζεται άμεσα από την Σ1, ανατρέπει την υπόθεση της Σ1 σχετικά με την τιμή του Α, πάνω στην οποία στηρίζονται οι επόμενες ενέργειες της Σ1.

2.4.3 Επανεγγραφή μη-ολοκληρωμένων δεδομένων (Συγκρούσεις WW)

Η τρίτη περίπτωση ανώμαλης συμπεριφοράς είναι όταν μια συναλλαγή Σ2 θα μπορούσε να επανεγγράψει την τιμή ενός αντικειμένου Α, ενώ αυτή έχει ήδη τροποποιηθεί από μια συναλλαγή Σ1 που είναι ακόμα ενεργή. Ακόμα και αν η Σ2 δε διαβάσει την τιμή του Α που έχει γραφεί από την Σ1, υπάρχει ένα πιθανό πρόβλημα, όπως δείχνει το παρακάτω παράδειγμα.

Ας θεωρηθεί η περίπτωση δύο υπαλλήλων, των οποίων οι μισθοί πρέπει να είναι ίσοι. Η συναλλαγή Σ1 δίνει στους μισθούς την τιμή €1000, και η συναλλαγή Σ2 τους δίνει την τιμή €2000.

Αν εκτελεστούν οι συναλλαγές αυτές σειριακά, με την Σ1 να ακολουθείται από την Σ2, τότε ο μισθός των υπαλλήλων είναι €2000. Η σειριακή εκτέλεση με την Σ2 να ακολουθείται από την Σ1 δίνει μισθό €1000. Τα δύο αυτά σενάρια είναι αποδεκτά όσον αφορά τη συνέπεια της βάσης. η παρατήρηση έγκειται στο γεγονός, πως καμιά συναλλαγή δε διαβάζει την τιμή του μισθού πριν τον τροποποιήσει. Αυτό ονομάζεται τυφλή εγγραφή για προφανείς λόγους.

Το τελευταίο σενάριο αποτελεί την ακόλουθη εναλλασσόμενη εκτέλεση των ενεργειών των συναλλαγών Σι και Σ2, που είναι η εξής: η Σι περιγράφει το μισθό του πρώτου υπαλλήλου €1000, η Σ2 περιγράφει το μισθό του δεύτερου €2000, η Σ1 περιγράφει το μισθό του δεύτερου €1000, και τέλος η Σ2 περιγράφει το μισθό του πρώτου €2000. Το αποτέλεσμα δεν είναι ταυτόσημο με το αποτέλεσμα καμιάς από τις δύο δυνατές σειριακές εκτελέσεις, και συνεπώς το εναλλασσόμενο χρονοπρόγραμμα δεν είναι σειριοποιήσιμο. Καταστρατηγεί το απαιτούμενο κριτήριο συνέπειας ότι οι δύο μισθοί πρέπει να είναι ίσοι.

3 ^{ο} ΚΕΦΑΛΑΙΟ

ΕΛΕΓΧΟΣ ΤΑΥΤΟΧΡΟΝΙΣΜΟΥ ΜΕ ΚΛΕΙΔΩΜΑΤΑ

3.1 Εισαγωγή

Στο κεφάλαιο αυτό, περιγράφεται πιο αναλυτικά ο έλεγχος ταυτοχρονισμού. Συγκεκριμένα παρουσιάζονται τα πρωτόκολλα κλειδώματος και πως αυτά εγγυώνται τις διάφορες σημαντικές ιδιότητες των χρονοπρογραμμάτων της προηγούμενης ενότητας. Στη συνέχεια, αναλύονται τα πρωτόκολλα κλειδώματος δύο φάσεων.

3.2 Μηχανισμοί κλειδώματος

Ένας από τους βασικότερους μηχανισμούς συντονισμού της εκτέλεσης ενός συνόλου συναλλαγών στηρίζεται στη χρήση κλειδωμάτων (locks). Το κλείδωμα αντιστοιχεί σε ένα τμήμα δεδομένων και πρόκειται για μία μεταβλητή που περιγράφει την κατάσταση του τμήματος των δεδομένων σε σχέση με τις λειτουργίες που ενεργούν στα συγκεκριμένα δεδομένα. Δύο είναι οι βασικοί τύποι κλειδωμάτων που χρησιμοποιούνται:

- **κλείδωμα ανάγνωσης (readlock) ή διαμοιραζόμενο κλειδί (sharedlock)** που επιτρέπει σε μία συναλλαγή να διαβάσει τα δεδομένα, αλλά όχι να τα ενημερώσει ή να τα διαγράψει,

- **κλείδωμα αποθήκευσης (writelock) ή αποκλειστικό κλειδί (exclusivelock)** που επιτρέπει σε μία συναλλαγή να διαβάσει ή/και να ενημερώσει τα δεδομένα.

Κάθε φορά που μία συναλλαγή θέλει να προσπελάσει μία σελίδα δεδομένων ζητά να της χορηγηθεί ένα κλείδωμα. Επειδή, η ανάγνωση της ίδιας σελίδας δεδομένων από πολλές συναλλαγές δε δημιουργεί πρόβλημα, επιτρέπεται να χορηγηθούν πολλά κλειδώματα ανάγνωσης σε πολλές συναλλαγές.

Αντίθετα, ένα κλείδωμα αποθήκευσης μπορεί να χορηγηθεί μόνο σε μία συναλλαγή, διότι δεν επιτρέπεται σε πολλές συναλλαγές να ενημερώνουν ταυτόχρονα την ίδια σελίδα δεδομένων. Έτσι, για όσο διάστημα μία συναλλαγή έχει κλείδωμα αποθήκευσης για μία σελίδα δεδομένων, δεν επιτρέπεται η χορήγηση κλειδώματος σε άλλες συναλλαγές.

Ο μηχανισμός κλειδωμάτων λειτουργεί ως εξής:

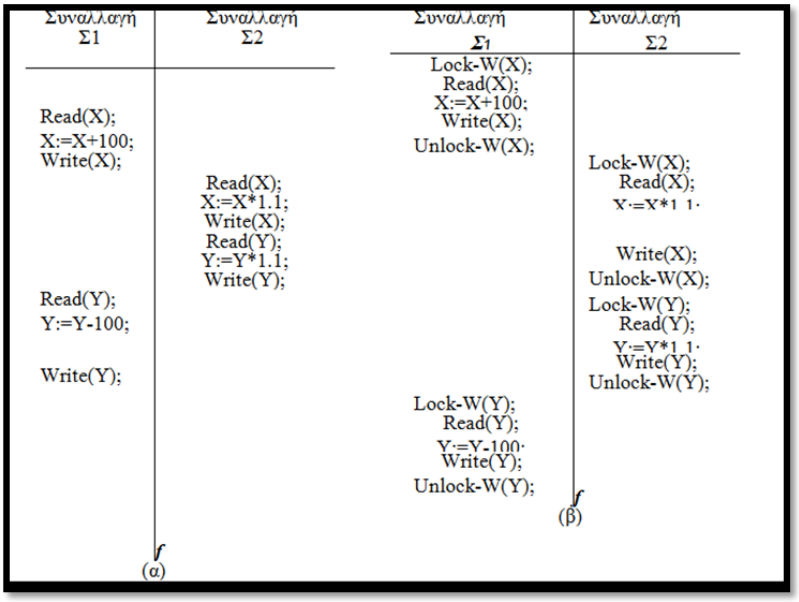
- Μία συναλλαγή που απαιτεί ανάγνωση δεδομένων ζητά από το σύστημα να της χορηγηθεί ένα κλείδωμα ανάγνωσης. Αν απαιτείται ενημέρωση δεδομένων, τότε ζητά κλείδωμα αποθήκευσης.
- Αν δεν υπάρχει άλλο κλείδωμα για τη συγκεκριμένη σελίδα δεδομένων, τότε το κλείδωμα χορηγείται στη συναλλαγή.
- Αν υπάρχει, ήδη, άλλο κλείδωμα που έχει χορηγηθεί σε άλλη συναλλαγή, τότε πραγματοποιείται έλεγχος για το αν θα χορηγηθεί το κλείδωμα στη νέα συναλλαγή ή όχι. Αν η σελίδα δεδομένων έχει κλειδωθεί με χρήση κλειδώματος ανάγνωσης και η νέα συναλλαγή ζητά επίσης κλείδωμα ανάγνωσης, τότε το σύστημα επιτρέπει τη χορήγηση του κλειδώματος. Σε διαφορετική περίπτωση η νέα συναλλαγή πρέπει να περιμένει μέχρι η προηγούμενη συναλλαγή ολοκληρώσει την εργασία με τη σελίδα δεδομένων και απελευθερώσει κλείδωμα.

- Μία συναλλαγή απελευθερώνει το κλείδωμα, όταν η εκτέλεση της ολοκληρωθεί με επιτυχία είτε με αποτυχία. Οι μόνιμες αλλαγές στα δεδομένα είναι ορατές από τις υπόλοιπες συναλλαγές, μόνο μετά την απελευθέρωση του κλειδώματος αποθήκευσης από τη συναλλαγή.

Μερικά συστήματα επιτρέπουν σε μία συναλλαγή να αλλάξει τον τύπο του κλειδώματος που της έχει χορηγηθεί. Θεωρείστε μία συναλλαγή που έχει ένα κλείδωμα ανάγνωσης σε σελίδα δεδομένων. Αν αργότερα η συναλλαγή πρέπει να ενημερώσει κάποια δεδομένα, μπορεί να ζητήσει αναβάθμιση (upgrade) του κλειδώματος από κλείδωμα ανάγνωσης σε κλείδωμα αποθήκευσης. Αντίθετα, αν η εγγραφή των δεδομένων έχει ολοκληρωθεί, αλλά η συναλλαγή δεν έχει ολοκληρώσει την ανάγνωση των δεδομένων, τότε μπορεί να ζητήσει υποβάθμιση (downgrade) του κλειδώματος από αποθήκευσης σε ανάγνωσης. Ο μηχανισμός κλειδώματος, όπως αναλύθηκε προηγουμένως, δε δίνει πάντοτε λύση στο πρόβλημα της συνέπειας των δεδομένων.

Για παράδειγμα, ας θεωρηθεί το χρονοπρόγραμμα εκτέλεσης της Εικόνας 15(α). Προσθέτοντας τις κατάλληλες εντολές Lock και Unlock παίρνουμε το χρονοπρόγραμμα στην Εικόνα 1.5.1(β). Έστω ότι οι αρχικές τιμές για τα δεδομένα X και Y είναι X = 100 και Y=400. Αν εκτελεσθεί πρώτα η συναλλαγή Σ1 και μετά η Σ2 θα έχουμε στο τέλος της εκτέλεσης και των δύο συναλλαγών X=220 και Y=330. Αν η Σ2 εκτελεσθεί πριν τη Σ1 τότε θα έχουμεX=210 και Y=340. Όμως, η εκτέλεση του χρονοπρογράμματος του παραδείγματος δίνει X=220 και Y=340. Αυτό σημαίνει ότι το χρονοπρόγραμμα δεν είναι σειριοποιήσιμο.

Το πρόβλημα εντοπίζεται στον τρόπο δέσμευσης και απελευθέρωσης των κλειδιών από τις συναλλαγές. Η αυθαίρετη δέσμευση ή απελευθέρωση κλειδιών μπορεί να δημιουργήσει σοβαρά προβλήματα στη συνέπεια των δεδομένων.



Εικόνα 16 - Παράδειγμα χρήσης μηχανισμού κλειδώματος

3.3 Πρωτόκολλα κλειδώματος - Αυστηρό 2PL

Ένας μηχανισμός κλειδώματος πρέπει να πληροί ορισμένους κανόνες που να εγγυώνται τη συνέπεια των δεδομένων. Οι κανόνες αυτοί ορίζουν ένα πρωτόκολλο κλειδώματος (locking protocol) και εγγυώνται ότι το χρονοπρόγραμμα που προκύπτει είναι σειριοποιήσιμο. Στη συνέχεια εξετάζουμε δύο από τα βασικότερα πρωτόκολλα κλειδώματος (Raz, 1992).

Ένα ΣΔΒΔ πρέπει να μπορεί να εξασφαλίσει ότι επιτρέπονται μόνο σειριοποιήσιμα, επαναφέρσιμα χρονοπρογράμματα και ότι δε χάνονται ενέργειες ολοκληρωμένων συναλλαγών, κατά την αναίρεση

εγκαταλειμμένων συναλλαγών. Συνήθως, ένα ΣΔΒΔ χρησιμοποιεί ένα πρωτόκολλο κλειδώματος για να επιτύχει κάτι τέτοιο.

Ένα **πρωτόκολλο κλειδώματος** είναι ένα σύνολο κανόνων που πρέπει να ακολουθεί κάθε συναλλαγή (και να επιβάλλεται από το ΣΔΒΔ), ώστε, ακόμα και αν οι ενέργειες πολλών συναλλαγών εκτελούνται με εναλλασσόμενο τρόπο, να διασφαλίζεται ότι το τελικό αποτέλεσμα είναι ταυτόσημο με αυτό της εκτέλεσης όλων των συναλλαγών με κάποια συγκεκριμένη σειρά. Το πιο συνηθισμένο πρωτόκολλο κλειδώματος, ονομάζεται Αυστηρό Κλείδωμα Δύο Φάσεων, ή Αυστηρό 2PL (2-Phase Locking protocol), και ορίζει δύο κανόνες.

Ο πρώτος κανόνας περιγράφει ότι: Αν μια συναλλαγή Σ1 θέλει να διαβάσει (αντίστοιχα να τροποποιήσει) ένα αντικείμενο, πρώτα ζητά ένα κοινόχρηστο (αντίστοιχα ένα αποκλειστικό) κλείδωμα πάνω στο αντικείμενο.

Βέβαια, μια συναλλαγή που έχει ένα αποκλειστικό κλείδωμα μπορεί και να διαβάσει το αντικείμενο και δεν απαιτείται ένα επιπλέον κοινόχρηστο κλείδωμα. Μια συναλλαγή που ζητά ένα κλείδωμα αναστέλλεται, μέχρι να μπορέσει το ΣΔΒΔ να χορηγήσει το ζητούμενο κλείδωμα. Το ΣΔΒΔ καταγράφει τα κλειδώματα που έχει χορηγήσει και διασφαλίζει πως αν μια συναλλαγή κρατά ένα αποκλειστικό κλείδωμα σε ένα αντικείμενο, καμιά άλλη συναλλαγή δεν κρατά ένα κοινόχρηστο ή αποκλειστικό κλείδωμα στο ίδιο αντικείμενο.

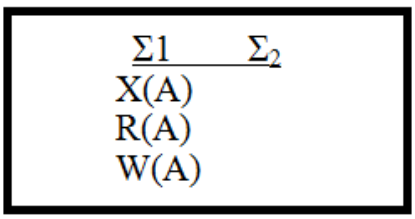
Ο δεύτερος κανόνας του Αυστηρού 2PL περιγράφει ότι: Όλα τα κλειδώματα που κρατούνται από μια συναλλαγή αποδεσμεύονται με την ολοκλήρωση της συναλλαγής.

Οι αιτήσεις απόκτησης και αποδέσμευσης κλειδωμάτων εισάγονται σε μια συναλλαγή αυτόματα από το ΣΔΒΔ. Οι χρήστες δεν χρειάζεται να ασχολούνται με αυτές τις λεπτομέρειες.

Ουσιαστικά, το πρωτόκολλο κλειδώματος επιτρέπει μόνο «ασφαλείς» εναλλασσόμενες εκτελέσεις των συναλλαγών. Αν δύο συναλλαγές προσπελαύνουν τελείως ανεξάρτητα μέρη της βάσης, είναι δυνατό να αποκτήσουν ταυτόχρονα τα κλειδώματα που χρειάζονται.

Αντίθετα, αν δύο συναλλαγές προσπελαύνουν το ίδιο αντικείμενο, και μια από αυτές θέλει να το τροποποιήσει, στην πραγματικότητα οι ενέργειες τους ταξινομούνται σειριακά - όλες οι ενέργειες της μιας συναλλαγής (αυτής που παίρνει το κλείδωμα στο αντικείμενο πρώτη) ολοκληρώνονται πριν (το κλείδωμα αποδεσμευτεί και) η άλλη συναλλαγή μπορέσει να συνεχίσει.

Συμβολίζουμε την ενέργεια μιας συναλλαγής Σ να ζητήσει ένα κοινόχρηστο (αντίστοιχα αποκλειστικό) κλείδωμα σε ένα αντικείμενο Ο με $Ξ_Σ$(ü) (αντίστοιχα με $X_Σ$(Ο)), και παραλείπουμε τον δείκτη που δηλώνει τη συναλλαγή όταν αυτό προκύπτει από τα συμφραζόμενα. Για παράδειγμα, θεωρείστε το χρονοπρόγραμμα όπου: η Σι θα άλλαζε το Α από 10 σε 20, έπειτα η Σ2 (που διαβάζει την τιμή 20 για το Α) θα άλλαζε το Β από 100 σε 200, και τέλος η Σι θα διάβαζε την τιμή 200 για το Β. Αν όμως εκτελούνταν σειριακά, είτε η Σι είτε η Σ2 θα εκτελούνταν πρώτη και θα διάβαζε τις τιμές 10 για το Α, 4 και 100 για το Β: προφανώς η εναλλασσόμενη εκτέλεση δεν είναι ταυτόσημη με καμιά σειριακή εκτέλεση.

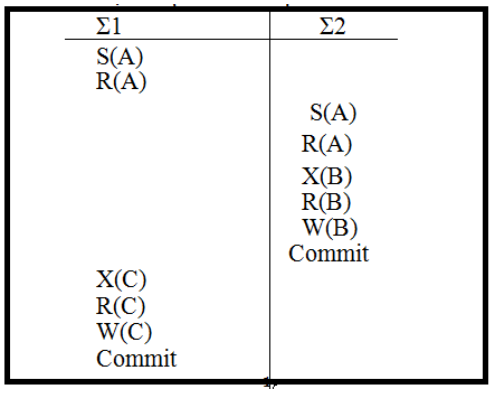


Εικόνα 17 - Χρονοπρόγραμμα που περιγράφει το Αυστηρό πρωτόκολλο 2PL

Αν χρησιμοποιηθεί το Αυστηρό πρωτόκολλο 2PL η παραπάνω εναλλασσόμενη εκτέλεση δεν επιτρέπεται. Ας υποθέσουμε πως οι

συναλλαγές εκτελούνται με την ίδια σχετική ταχύτητα όπως στο παράδειγμα. Η Σ1 θα αποκτούσε πρώτα ένα αποκλειστικό κλείδωμα στο Α και έπειτα θα το διάβαζε και θα το έγραφε (Εικόνα 17). Μετά, η Σ2 θα ζητούσε ένα κλείδωμα στο Α. Όμως, αυτή η αίτηση δεν μπορεί να ικανοποιηθεί έως ότου η Σ1 αποδεσμεύσει το αποκλειστικό κλείδωμα στο Α, οπότε το ΣΔΒΔ αναστέλλει την Σ2. Έτσι, η Σ1 προχωρά και αποκτά ένα αποκλειστικό κλείδωμα στο Β, και τελικά ολοκληρώνεται οπότε αποδεσμεύει τα κλειδώματά της. Τώρα, η αίτηση της Σ2 ικανοποιείται και η Σ2 συνεχίζει την εκτέλεση της. Σε αυτό το παράδειγμα η χρήση του πρωτοκόλλου κλειδώματος οδηγεί σε μια σειριακή εκτέλεση των συναλλαγών.

Γενικά, όμως, οι ενέργειες διαφορετικών συναλλαγών θα μπορούσαν να εκτελούνται εναλλασσόμενα. Ως παράδειγμα, μπορεί να ληφθεί η εναλλαγή των συναλλαγών που παρουσιάζονται στην Εικόνα 18, και η οποία επιτρέπεται από το Αυστηρό πρωτόκολλο 2PL.



Εικόνα 18 - Εναλλαγή των ενεργειών με το Αυστηρό 2PL πρωτόκολλο

3.4 Πρωτόκολλο κλειδώματος βασισμένο σε γράφημα

Αν δεν υπάρχουν διαθέσιμες πληροφορίες σχετικά με τη σειρά πραγματοποίησης των προσπελάσεων των δεδομένων από κάθε συναλλαγή, τότε η χρήση του πρωτοκόλλου κλειδώματος δύο φάσεων κρίνεται

απαραίτητη για την παραγωγή ενός σειριοποιήσιμου ως προς τις συγκρούσεις χρονοδιαγράμματος. Ωστόσο, σε πολλές περιπτώσεις υπάρχουν περισσότερες πληροφορίες για τον τρόπο προσπέλασης των δεδομένων και μπορούμε να χρησιμοποιήσουμε την πληροφορία αυτή για την παραγωγή ενός σειριοποιήσιμου χρονοδιαγράμματος χωρίς να χρησιμοποιηθεί πρωτόκολλο δύο φάσεων. Η πιο απλή μορφή πληροφορίας που μπορούμε να εξάγουμε είναι η σειρά πραγματοποίηση των προσπελάσεων των δεδομένων από τις συναλλαγές.

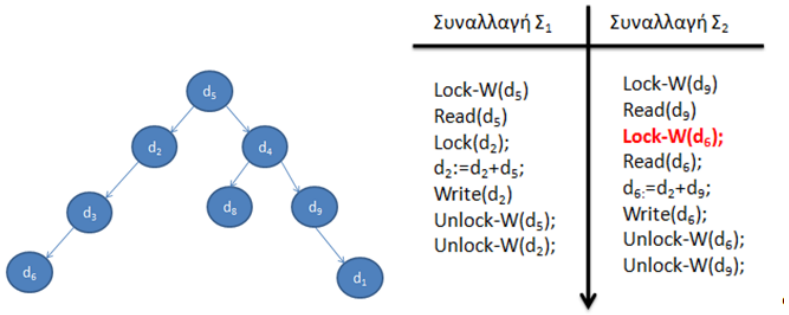
Έστω D={ d_1, …, d_n} ένα σύνολο n δεδομένων που προσπελαύνονται από τις συναλλαγές είτε για ανάγνωση είτε για αποθήκευση. Ορίζουμε μια μερική διάταξη στο σύνολο D ως εξής: τα δεδομένα d_i και d_j συνδέονται με τη σχέση => αν κάθε συναλλαγή που προσπελαύνει τα d_i και d_j προσπελαύνει το d_i πριν το d_j. Η μερική διάταξη που ορίσθηκε επιτρέπει την αναπαράσταση του συνόλου ως κατευθυνόμενο άκυκλο γράφημα (directed acyclic graph). Ειδική περίπτωση του κατευθυνόμενου άκυκλου γραφήματος είναι το δενδρικό γράφημα, όπου εφαρμόζεται το δενδρικό πρωτόκολλο (tree protocol) που υποστηρίζει μόνο κλειδαριές ανάγνωσης.

Το δενδρικό πρωτόκολλο έχει τους εξής κανόνες:

- Η κάθε συναλλαγή μπορεί να ζητήσει χορήγηση της πρώτης κλειδαριάς σε οποιοδήποτε δεδομένο,
- Μια συναλλαγή μπορεί να ζητήσει τη χορήγηση κλειδαριάς στο δεδομένο dj μόνο αν έχει ήδη κλειδώσει το δεδομένο d_i και ισχύει d_i=>d_j,
- Μια συναλλαγή μπορεί να απελευθερώσει μια κλειδαριά οποιαδήποτε χρονική στιγμή,

- Ένα δεδομένο που έχει κλειδωθεί από μια συναλλαγή και στη συνέχεια η κλειδαριά έχει απελευθερωθεί, δεν επιτρέπεται να κλειδωθεί για δεύτερη φορά από την ίδια συναλλαγή.

Αποδεικνύεται ότι κάθε χρονοδιάγραμμα που ικανοποιεί τους κανόνες του δενδρικού πρωτοκόλλου είναι σειριοποιήσιμο ως προς τις συγκρούσεις. Στην παρακάτω εικόνα βλέπουμε ένα παράδειγμα δενδρικού κατευθυνόμενου γράφου και δύο συναλλαγές εκ των οποίων η Σ2 παραβιάζει το δεύτερο κανόνα του δενδρικού πρωτοκόλλου.



Εικόνα 19 - Παράδειγμα εφαρμογής του δενδρικού πρωτοκόλλου κλειδώματος

3.5 Διαχείριση αδιεξόδου

Οι μηχανισμοί ελέγχου ταυτόχρονων προσπελάσεων οι οποίοι βασίζονται σε πρωτόκολλα κλειδώματος ενδέχεται να φέρουν το σύνολο των συναλλαγών σε αδιέξοδο (deadlock). Ένα σύνολο συναλλαγών βρίσκεται σε αδιέξοδο όταν καμία συναλλαγή δεν είναι εφικτό να συνεχίσει την εκτέλεσή της, λόγω αναμονής για την απελευθέρωση κλειδαριών. Αν όλες οι συναλλαγές αναμένουν την απελευθέρωση κάποιας κλειδαριάς αυτό δεν πρόκειται να συμβεί ποτέ, με αποτέλεσμα καμία συναλλαγή να μην μπορεί να συνεχίσει την εκτέλεσή της (Padua,2009). Το αδιέξοδο αποτελεί παθολογική

κατάσταση και πρέπει να αποφεύγεται διότι συνήθως οδηγεί σε αναγκαστικό τερματισμό μιας συναλλαγής έτσι ώστε οι υπόλοιπες να συνεχίσουν την εκτέλεσή τους. Η διαχείριση του αδιεξόδου πραγματοποιείται γενικά με δύο μεθόδους:

- **Αποφυγή αδιεξόδου**, όπου το σύστημα ελέγχει την πιθανότητα να συμβεί αδιέξοδο και προσπαθεί να το αποφύγει,
- **Αναγνώριση αδιεξόδου και επανάκτηση** (deadlock detection and recovery) όπου το σύστημα είναι σε θέση να αναγνωρίσει ένα αδιέξοδο και προβαίνει σε ενέργειες για την κατάργησή του.

Αν σε ένα σύστημα τα αδιέξοδα συμβαίνουν συχνά τότε συνιστάται η πρώτη μέθοδος. Οι έλεγχοι που εκτελεί το σύστημα για την αποφυγή του αδιεξόδου είναι χρονοβόροι.

3.6 Αποφυγή αδιεξόδου

Μια απλή μέθοδος που χρησιμοποιείται για την αποφυγή του αδιεξόδου σε ένα σύνολο συναλλαγών είναι ο ορισμός ενός μέγιστου χρονικού διαστήματος για την αναμονή χορήγησης μιας κλειδαριάς. Αν εξαντληθεί το χρονικό διάστημα και η κλειδαριά δεν έχει ακόμη χορηγηθεί στη συναλλαγή, το σύστημα τερματίζει τη διαδικασία και την επαναδημιουργεί. Η απλή αυτή μέθοδος αποφυγής αδιεξόδου χρησιμοποιείται από πολλά ΣΔΒΔ λόγω της απλότητάς της.

Μια άλλη εναλλακτική λύση στο πρόβλημα της αποφυγής αδιεξόδου στηρίζεται σε χρονικές σφραγίδες (timestamps). Παρατίθενται δύο αλγόριθμοι (Rosenkrantz,1978):

- **Wait-Die:** Σύμφωνα με αυτή τη μέθοδο επιτρέπει σε μια παλαιότερη συναλλαγή (μικρή χρονική σφραγίδα) να περιμένει την απελευθέρωση μιας κλειδαριάς από μία νεότερη (μεγαλύτερη χρονική

σφραγίδα). Διαφορετικά, η συναλλαγή τερματίζεται και το σύστημα επανεκκινεί.

- **Wound-Wait:** Επιτρέπεται σε μια νεότερη συναλλαγή να περιμένει την εκτέλεση μίας παλαιότερης. Αν η παλαιότερη συναλλαγή ζητά τη χορήγηση μιας κλειδαριάς που είναι δεσμευμένη από μία νεότερη συναλλαγή, τότε η νεότερη συναλλαγή τερματίζεται.

3.7 Αναγνώριση και κατάργηση αδιεξόδου

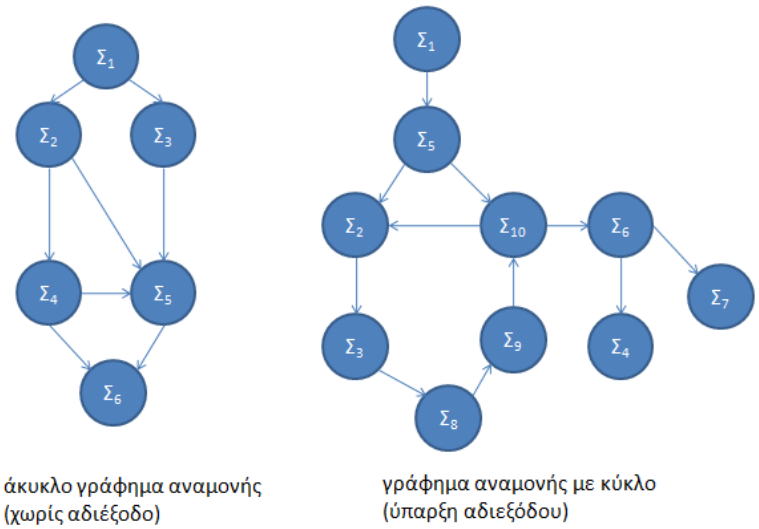
Το σύστημα πρέπει να είναι σε θέση να γνωρίζει αν ένα σύνολο συναλλαγών βρίσκεται σε αδιέξοδο. Ειδικά αν το ΣΔΒΔ δε διαθέτει μέθοδο αποφυγής αδιεξόδου, η αναγνώριση αδιεξόδου θεωρείται απαραίτητη. Μετά την αναγνώριση ενός αδιεξόδου το σύστημα πρέπει να επαναφέρει το σύνολο συναλλαγών σε μια κατάσταση όπου να μπορεί να συνεχισθεί η εκτέλεσή τους.

Η αναγνώριση ενός αδιεξόδου μπορεί να διατυπωθεί με την αναπαράσταση του συνόλου των συναλλαγών με τη βοήθεια κατευθυνόμενου γράφου που καλείται γράφημα αναμονής (wait graph). Ο γράφος αυτός αποτελείται από ένα σύνολο κορυφών V και ένα σύνολο ακμών Ε. Για κάθε συναλλαγή υπάρχει και μία αντίστοιχη κορυφή στο γράφο. Αν στο γράφο υπάρχει ακμή που αρχίζει από την κορυφή Σi και τελειώνει στην κορυφή Σj, τότε αυτό σημαίνει ότι η συναλλαγή Σi αναμένει μέχρι η συναλλαγή Σj να απελευθερώσει την κλειδαριά σε ένα τμήμα δεδομένων που πρέπει να προσπελάσει η Σi.

Αν στο γράφο αναμονής δεν υπάρχει κύκλος, τότε δεν υπάρχει αδιέξοδο. Αντίθετα, η ύπαρξη έστω και ενός κύκλου σημαίνει ότι ένα υποσύνολο των συναλλαγών βρίσκεται σε αδιέξοδο. Στην παρακάτω εικόνα, παρατηρούμε δύο γράφους αναμονής όπου ο πρώτος δεν περιέχει κύκλο ενώ ο δεύτερος

γράφος περιέχει ένα κύκλο που προγραμματίζεται από τις κορυφές Σ_2, Σ_3, Σ_8, Σ_9 και Σ_{10}.

Εφόσον η ύπαρξη κύκλου στο γράφο αναμονής συνεπάγεται την ύπαρξη αδιεξόδου για ένα υποσύνολο συναλλαγών, αρκεί να πραγματοποιείται έλεγχος του γραφήματος ανά τακτά χρονικά διαστήματα. Αν συμβαίνουν συχνά αδιέξοδα τόσο επίσης συχνός πρέπει να είναι και ο έλεγχος του γράφου αναμονής. Το ΣΔΒΔ μπορεί να διατηρεί στατιστικά στοιχεία για τον αριθμό των αδιεξόδων και τη συχνότητα εμφάνισης τους ανάλογα να προσδιορίζεται και η συχνότητα ελέγχου για αδιέξοδο.



Εικόνα 20 - Παράδειγμα γραφήματος αναμονής χωρίς κύκλο και με κύκλο

Η συχνότερα χρησιμοποιούμενη μέθοδος είναι ο τερματισμός μιας ή περισσοτέρων συναλλαγών και η επαναφορά των δεδομένων στην κατάσταση που βρισκόταν πριν από την εκτέλεση των συναλλαγών αυτών (ROLLBACK). Η μέθοδος αυτή αποτελείται από τα εξής βήματα:

1. Επιλογή συναλλαγών για ROLLBACK. Το πρώτο βήμα της μεθόδου αφορά στην επιλογή μίας ή περισσοτέρων συναλλαγών για τις οποίες πρέπει να εκτελεσθεί η λειτουργία ROLLBACK. Η επιλογή των συναλλαγών πραγματοποιείται με βάση το ελάχιστο κόστος. Στον προσδιορισμό του κόστους συμμετέχουν οι ακόλουθοι παράγοντες:

    - Ο μέχρι στιγμής χρόνος εκτέλεσης της συναλλαγής και ο χρόνος που υπολείπεται για την ολοκλήρωση της συναλλαγής.
    - Ο αριθμός των δεδομένων που έχουν ενημερωθεί από τη συναλλαγή.
    - Ο αριθμός των δεδομένων που πρέπει να προσπελασθούν από τη συναλλαγή.
    - Ο αριθμός των συναλλαγών όπου θα εφαρμοσθεί η λειτουργία ROLLBACK.

2. Επιλογή σημείου ROLLBACK. Στο δεύτερο βήμα, για τις συναλλαγές που έχουν επιλεγεί πρέπει να προσδιορισθεί μέχρι ποιο σημείο θα εφαρμοσθεί η λειτουργία ROLLBACK. Η πιο απλή λύση είναι να γίνει επαναφορά της συναλλαγής στην αρχή, καταργώντας όλες τις αλλαγές που έχει επιφέρει στα δεδομένα. Η λύση αυτή πολλές φορές έχει μεγάλο κόστος, διότι η συναλλαγή μπορεί να έχει ενημερώσει είναι ένα μεγάλο αριθμό δεδομένων. Συνήθως, το αδιέξοδο καταργείται εφαρμόζοντας ROLLBACK στις τελευταίες εντολές της συναλλαγής.

3. Αποφυγή ατέρμονης αναμονής. Η ατέρμονη αναμονή είναι ένα φαινόμενο που μπορεί να εμφανισθεί αν επιλέγεται συνεχώς η ίδια συναλλαγή για την εφαρμογή του ROLLBACK. Αυτό

σημαίνει ότι συνεχώς η ίδια συναλλαγή εμποδίζεται να ολοκληρώσει την εκτέλεσή της. Το ΣΔΒΔ μπορεί να αποτρέψει την επιλογή της ίδιας συναλλαγής περισσότερο από έναν αριθμό επιλογών, δίνοντας έτσι την ευκαιρία στη συναλλαγή να ολοκληρώνει την επεξεργασία της.

4 Ο ΚΕΦΑΛΑΙΟ

ΕΠΑΝΑΦΟΡΑ ΔΕΔΟΜΕΝΩΝ ΜΕΤΑ ΑΠΟ ΚΑΤΑΡΡΕΥΣΗ ΣΥΣΤΗΜΑΤΟΣ

4.1 Εισαγωγή

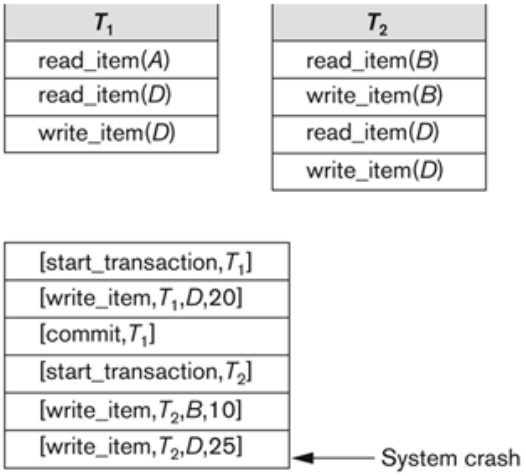
Η επαναφορά δεδομένων αποτελεί μια σημαντική λειτουργία του Συστήματος Βάσης Δεδομένων, η οποία απαιτείται σε περιπτώσεις βλάβης του συστήματος που προκαλούν προσωρινή διακοπή λειτουργίας (π.χ. απότομη διακοπή συστήματος).

Οι συχνότερες αιτίες που οδηγούν σε διακοπή της λειτουργίας του συστήματος είναι:

- **Πτώση συστήματος**, λόγω εσφαλμένης λειτουργίας του υλικού ή του λογισμικού συστήματος.
- **Πρόβλημα στο μέσο αποθήκευσης**, το οποίο μπορεί να οφείλεται σε καταστροφή της μαγνητικής επιφάνειας του δίσκου, του μηχανικού ή του ηλεκτρονικού υλικού του δίσκου.
- **Σφάλμα λογισμικού εφαρμογής**, το οποίο οφείλεται συνήθως σε σφάλμα εκτέλεσης (run-time error) κατά τη διάρκεια ενημέρωσης των δεδομένων της ΒΔ.
- **Φυσικά αίτια**, όπως πυρκαγιές, σεισμοί κτλ.

Οποιοδήποτε και αν είναι το αίτιο, η απότομη διακοπή της λειτουργίας του ΣΔΒΔ πρέπει να ακολουθείται από τη λειτουργία επανάκτησης, έτσι ώστε ο

κίνδυνος απώλειας να περιοριστεί στο ελάχιστο. Στην εικόνα 21 ακολουθεί ένα παράδειγμα επαναφοράς, όπου δύο συναλλαγές χρησιμοποιούν τις λειτουργίες ανάγνωσης και εγγραφής. Παρατηρείται το σημείο κατάρρευσης και οι εγγραφές της T2 αγνοούνται κατά την επαναφορά συστήματος, ενώ οι ενέργειες [write_item, ,] της T1 επαναλαμβάνονται.



Εικόνα 21 - Παράδειγμα επαναφοράς

4.2 Απαιτήσεις επανάκτησης

Για να είναι εφικτή η λειτουργία της επανάκτησης δεδομένων το ΣΔΒΔ πρέπει να διαθέτει:

- **Μηχανισμό δημιουργίας αντιγράφων ασφαλείας** (backup) για τα δεδομένα της ΒΔ,
- **Καταγραφή ημερολογίου** (log) όπου να αποθηκεύεται η κατάσταση της κάθε συναλλαγής και οι αλλαγές που έχουν πραγματοποιηθεί στη ΒΔ.

- **Μηχανισμό χρονικών σημείων ελέγχου** (checkpoints) τα οποία σηματοδοτούν τις στιγμές που οι αλλαγές που έχουν επιφέρει οι συναλλαγές έχουν γίνει μόνιμες.
- **Μέθοδο επανάκτησης**, η οποία λαμβάνοντας υπόψη τα αντίγραφα ασφαλείας, τις πληροφορίες ημερολογίου και τα χρονικά σημεία ελέγχου, επαναφέρει τη ΒΔ σε κατάσταση συνέπειας.

Ο μηχανισμός παραγωγής αντιγράφων ασφαλείας πρέπει να λειτουργεί χωρίς να απαιτείται η διακοπή της λειτουργίας του συστήματος. Σε πολλές περιπτώσεις η λειτουργία του συστήματος πρέπει να είναι συνεχής και σε καμία περίπτωση δεν πρέπει να διακοπεί (π.χ. συστήματα ιατρικών δεδομένων, ή συστήματα που υποστηρίζουν στρατιωτικές εφαρμογές). Συνήθως ορίζεται κάποιο χρονικό διάστημα μετά την πάροδο του οποίου τα δεδομένα της ΒΔ αντιγράφονται σε άλλο χώρο αποθήκευσης. Επειδή τα αντίγραφα ασφαλείας καταλαμβάνουν σημαντικό χώρο στο δίσκο, συνήθως μεταφέρονται σε συσκευές μαγνητικών ταινιών. Η πιο αποδοτική, από άποψη χρόνου, μέθοδος δημιουργίας αντιγράφων ασφαλείας είναι η λεγόμενη αυξητική κατά την οποία αντιγράφονται τα δεδομένα που έχουν μεταβληθεί από το προηγούμενο αντίγραφο ασφαλείας. Έτσι αποφεύγεται η εξ ολοκλήρου χρονοβόρα αντιγραφή των δεδομένων τη ΒΔ.

Το ΣΔΒΔ διατηρεί ημερολόγιο όπου καταγράφονται διάφορα γεγονότα η γνώση των οποίων κρίνεται απαραίτητη. Στο ημερολόγιο αποθηκεύονται πληροφορίες σχετικά με εισαγωγές, διαγραφές και ενημερώσεις δεδομένων καθώς επίσης και τα χαρακτηριστικά των συναλλαγών που προκάλεσαν αυτές τις μεταβολές. Σε μεγάλα συστήματα που υποστηρίζουν μεγάλο αριθμό συναλλαγών καθημερινά (π.χ. τραπεζικά συστήματα) το αρχείο ημερολογίου μπορεί να είναι αρκετά μεγάλο. Το ημερολόγιο είναι ένα σημαντικό βοήθημα για το ΣΔΒΔ και μπορεί να χρησιμοποιηθεί για τον έλεγχο της απόδοσης του συστήματος.

Ένα χρονικό σημείο ελέγχου προσδιορίζει τη χρονική στιγμή εκτέλεσης των εξής ενεργειών:

- Το αρχείο ημερολογίου αντιγράφεται από την κύρια μνήμη στο δίσκο,
- Τα δεδομένα που είναι αποθηκευμένα στην απομονωτική μνήμη, αντιγράφονται στο δίσκο
- Στο αρχείο ημερολογίου αποθηκεύονται οι πληροφορίες που αφορούν στις συναλλαγές που είναι ενεργές τη συγκεκριμένη χρονική στιγμή.

Η χρήση των χρονικών σημείων ελέγχου βοηθά το ΣΔΒΔ στην αποδοτικότερη επανάκτηση καθώς προσδιορίζονται εύκολα οι συναλλαγές που πιθανόν να δημιουργήσουν ασυνέπεια μετά από απότομη διακοπή του συστήματος. Σε κάποιες από αυτές τις συναλλαγές πρέπει να εφαρμοσθεί η λειτουργία ROLLBACK και σε κάποιες άλλες η λειτουργία ROLLFORWARD. Η πρώτη λειτουργία επιφέρει κατάργηση των αλλαγών που έχε επιφέρει η συναλλαγή. Αντίθετα, η δεύτερη λειτουργία επαναλαμβάνει την εκτέλεση μιας συναλλαγής που δεν ολοκληρώθηκε λόγω διακοπής της λειτουργίας του συστήματος.

4.3 Μέθοδοι επαναφοράς

Η μέθοδος επαναφοράς που εφαρμόζεται κάθε φορά εξαρτάται από το μέγεθος της καταστροφής που έχει υποστεί η ΒΔ. Διακρίνουμε δύο περιπτώσεις:

1. Η πρώτη περίπτωση αφορά σε σημαντική απώλεια των δεδομένων της ΒΔ, που μπορεί να οφείλεται σε βλάβη του δίσκου του συστήματος. Στην περίπτωση αυτή απαιτείται η επανάκτηση των δεδομένων χρησιμοποιώντας το πλέον πρόσφατο αντίγραφο ασφαλείας. Στη συνέχεια χρησιμοποιείται το αρχείο ημερολογίου και εφαρμόζεται η λειτουργία ROLLFORWARD για τις συναλλαγές που

προκάλεσαν μεταβολή στα δεδομένα της ΒΔ μετά το τελευταίο χρονικό σημείο ελέγχου. Η μέθοδος αυτή προϋποθέτει ότι το αρχείο ημερολογίου δεν έχει υποστεί καταστροφή.

2. Η δεύτερη περίπτωση αφορά σε καταστάσεις ασυνέπειας δεδομένων χωρίς να συνοδεύονται από καταστροφή των δεδομένων της ΒΔ και του ημερολογίου. Για τις περιπτώσεις αυτές πρέπει να εφαρμοσθούν οι λειτουργίες ROLLBACK και ROLLFORWARD σε συγκεκριμένες συναλλαγές η εκτέλεση των οποίων έχει προκαλέσει την ασυνέπεια. Σημειώνεται ότι σε μία τέτοια περίπτωση δεν απαιτείται η χρήση αντιγράφου ασφαλείας. Η επανάκτηση μπορεί να πραγματοποιηθεί με τα δεδομένα της ΒΔ ως έχουν μετά την επανεκκίνηση του συστήματος και τις πληροφορίες του αρχείου ημερολογίου.

4.4 Ο αλγόριθμος ARIES

Ο ARIES (Algorithms for Recovery and Isolation Exploiting Semantics) αποτελεί έναν αλγόριθμο επαναφοράς που ο σχεδιασμός του έχει βασιστεί στο πρωτόκολλο WAL (Write Ahead Logging). Πρόκειται για έναν αλγόριθμο που λειτουργεί με μία μη-βίαιη στηλιτικά άποψη στις Βάσεις Δεδομένων (Mohan,1992).

Ο αλγόριθμος ARIES στηρίζεται σε τρεις αρχές (Mohan,1999):

- **Απευθείας εγγραφή κατά την καταγραφή** (Write ahead logging): Αν σημειωθεί κάποια αλλαγή σε ένα αντικείμενο που καταγράφεται για πρώτη φορά στο ημερολόγιο (log), τότε το ημερολόγιο πρέπει να αποθηκευθεί προτού εφαρμοστούν αλλαγές στο αντικείμενο κατά τη διάρκεια εγγραφής στο δίσκο.
- **Επανάληψη ιστορικού ενεργειών κατά τη διάρκεια της Επανάληψης** (Repeating history during Redo): Κατά την

επανεκκίνηση μετά από μια κατάρρευση του συστήματος, ο αλγόριθμος ARIES ανακτά τις ενέργειες της ΒΔ (πριν την κατάρρευση) και επαναφέρει το σύστημα στην τρέχουσα κατάσταση που βρισκόταν. Στη συνέχεια αναιρεί τις ενεργές συναλλαγές που λάμβαναν χώρο την ώρα της κατάρρευσης.

- **Καταγραφή αλλαγών κατά τη διάρκεια της αναίρεσης** (Logging changes during Undo): Οι αλλαγές που επιφέρονται στη βάση δεδομένων κατά τη διάρκεια της αναίρεσης καταγράφονται για να διασφαλιστεί ότι καμία ενέργεια δε θα επαναληφθεί σε περιπτώσεις επαναλαμβανόμενων επανεκκινήσεων.

4.4.1 Πρωτόκολλο WAL

Το πρωτόκολλο WAL (Write-Ahead Logging) ανήκει στην οικογένεια τεχνικών για την παροχή της ατομικότητας και της αντοχής (ACID ιδιότητες) σε Συστήματα Διαχείρισης Βάσεων Δεδομένων (Mohan,1992). Σε ένα σύστημα που χρησιμοποιεί το πρωτόκολλο WAL, όλες οι αλλαγές εγγράφονται σε ένα αρχείο καταγραφής, πριν την εφαρμογή τους. Συνήθως στο προαναφερθέν αρχείο εκχωρούνται και οι πληροφορίες που έχουν συλλεχθεί από τις φάσεις Επανάληψης και Αναίρεσης.

Προτού τα δεδομένα που βρίσκονται στην κύρια μνήμη εξαχθούν στη Βάση Δεδομένων (μη πτητική αποθήκευση), όλες οι εγγραφές του αρχείου που σχετίζονται με τα συγκεκριμένα δεδομένα πρέπει να εξαχθούν σε σταθερή αποθηκευτική μονάδα (συστήματα RAID). Ο κανόνας αυτός ονομάζεται κανόνας WAL. Κυριολεκτικά, ο κανόνας WAL απαιτεί μόνο ότι οι πληροφορίες της Αναίρεσης στο αρχείο καταγραφής έχουν εξαχθεί σε σταθερή αποθηκευτική μονάδα, και επιτρέπει την αποθήκευση των πληροφοριών σε μεταγενέστερο χρόνο. Η διαφορά είναι εμφανής σε συστήματα όπου οι πληροφορίες της Αναίρεσης και Επανάληψης αποθηκεύονται σε ξεχωριστά αρχεία καταγραφής.

Το πρωτόκολλο WAL εφαρμόζεται για να αποθηκεύσει τις εγγραφές του αρχείου καταγραφής στο αρχείο επίμονη πριν εγγραφούν τις σελίδες δεδομένων. Για παράδειγμα, ας ληφθεί υπόψη ένα πρόγραμμα που βρίσκεται στη μέση εκτέλεσης κάποιας λειτουργίας όταν η μηχανή βρίσκεται σε λειτουργία που χάνει δύναμη. Κατά την επανεκκίνηση, το πρόγραμμα αυτό θα απαιτούσε να γνωρίζει εάν η λειτουργία ολοκληρώθηκε με επιτυχία, ή κατά το ήμισυ ή απέτυχε. Εάν χρησιμοποιήθηκε ένα αρχείο WAL, το πρόγραμμα θα μπορούσε να ελέγξει αυτό το αρχείο καταγραφής και να συγκρίνει τι ήταν αυτό που έπρεπε να κάνει, όταν ξαφνικά έχασε την ενέργεια σε αυτό που έκανε. Με βάση αυτή τη σύγκριση, το πρόγραμμα θα μπορούσε να αποφασίσει να αναιρέσει αυτό που είχε ξεκινήσει, να ολοκληρώσει αυτό που είχε ξεκινήσει, ή να διατηρήσει τη διαδικασία ως είχε.

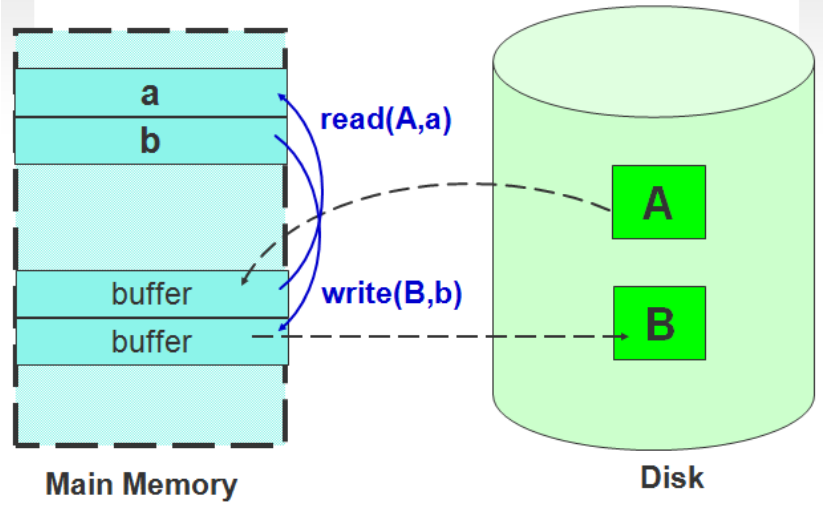
4.4.2 Καταγραφή

Για να εφαρμοσθεί ο αλγόριθμος ARIES απαιτείται ένας αριθμός από εγγραφές ημερολογίου να δημιουργηθούν κατά τη διάρκεια λειτουργίας της βάσης δεδομένων (Εικόνα 22). Συνήθως το καταγραφέν αρχείο ημερολογίου αποθηκεύεται σε «ασφαλή αποθήκευση», που είναι ένα μέσο αποθήκευσης που θεωρείται ότι επιβιώνει από καταρρεύσεις συστήματος και αστοχίες υλικού. Για να συγκεντρώσει τις απαραίτητες πληροφορίες για την καταγραφή απαιτούνται δύο δομές δεδομένων: ο πίνακας τροποποιημένων σελίδων (Dirty Page Table) και ο πίνακας συναλλαγών (Transaction Table).

Ο πίνακας τροποποιημένων σελίδων διατηρεί αρχείο όλων των σελίδων που έχουν τροποποιηθεί και δεν έχουν επανεγγραφεί στο δίσκο καθώς και τον πρώτο Αριθμό Ακολουθίας (Sequence Number) που μετατρέπει τη σελίδα σε βρώμικη. Ο συγκεκριμένος πίνακας διατηρεί αρχείο όλων των σελίδων που έχουν τροποποιηθεί και δεν έχουν επανεγγραφεί στο δίσκο καθώς και τον πρώτο Αριθμό Ακολουθίας (Sequence Number) που μετατρέπει τη σελίδα σε βρώμικη. Ο πίνακας συναλλαγών περιλαμβάνει όλες τις συναλλαγές που

τρέχουν και τον αριθμό ακολουθίας της πρόσφατης εγγραφής ημερολογίου που προκάλεσαν.

Δημιουργούνται αρχεία καταγραφής της μορφής (Αριθμός Ακολουθίας, ID Συναλλαγής, ID Σελίδας, Επανάληψη, Αναίρεση, Αριθμός Προηγούμενης Ακολουθίας). Τα πεδία Επανάληψη και Αναίρεση διατηρούν πληροφορίες σχετικά με τις αλλαγές που η εγγραφή ημερολογίου αποθηκεύει και πώς να τις αναιρεθούν. Ο Αριθμός Προηγούμενης Ακολουθίας είναι μια αναφορά στην προηγούμενη εγγραφή ημερολογίου που δημιουργήθηκε για τη συγκεκριμένη συναλλαγή. Σε περίπτωση συναλλαγής που είχε διακοπεί, είναι δυνατό να μεταφέρει το αρχείο καταγραφής με αντίστροφη σειρά με τη χρήση των αριθμών προηγούμενης ακολουθίας, αναιρώντας όλες τις ενέργειες που περιλαμβάνονται στο πλαίσιο της συγκεκριμένης συναλλαγής.



Εικόνα 22 - Οι διακεκομμένες γραμμές μπορεί να αναπαριστούν λειτουργίες που θα καθυστερήσουν ή να αγνοηθούν

Κάθε φορά που μια συναλλαγή ξεκινά ή εκτελείται γράφουμε την καταχώρηση «Begin Transaction» ή «End of Log» για την εν λόγω συναλλαγή, αντίστοιχα. Κατά τη διάρκεια της ανάκαμψης ή της αναίρεσης

(μιας συναλλαγής που ματαιώθηκε) ένας ιδιαίτερος τύπος ημερολογίου καταγραφής αποθηκεύεται, με την ονομασία Compensation Log Record ή Αρχείο Καταγραφής Αποζημίωσης, για να καταγράψει ότι η ενέργεια έχει ήδη αναιρεθεί. Τα αρχεία CLRs είναι της μορφής (Αριθμός Ακολουθίας, Αναγνωριστικό συναλλαγής, Αναγνωριστικό Σελίδας, Επανάληψη, Αριθμός της προηγούμενης ακολουθίας, Επόμενος Αριθμός Αναίρεσης Ακολουθίας). Το πεδίο Αναίρεση παραλείπεται, διότι οι εν λόγω πληροφορίες είναι ήδη αποθηκευμένες στο αρχικό ημερολόγιο καταγραφής για τις ενέργειες αυτές.

4.4.3 Επαναφορά

Η Επαναφορά αποτελείται από τρεις φάσεις.

Η πρώτη φάση, η Ανάλυση, λαμβάνει υπόψη όλες τις απαραίτητες πληροφορίες από το αρχείο καταγραφής.

Η φάση Επανάληψη επαναφέρει τη βάση δεδομένων στην ακριβή κατάσταση της συντριβής, συμπεριλαμβανομένων όλων των αλλαγών των μη επιβεβαιωμένων συναλλαγών που εκτελούνται εκείνη τη χρονική στιγμή.

Η φάση της Αναίρεσης, στη συνέχεια αναιρεί όλες τις μη επιβεβαιωμένες αλλαγές, αφήνοντας τη βάση δεδομένων σε μια σταθερή κατάσταση.

4.4.3.1 Ανάλυση

Κατά τη φάση της Ανάλυσης γίνεται αποκατάσταση του πίνακα βρώμικων σελίδων και τον πίνακα συναλλαγών που ίσχυαν κατά τη στιγμή της σύγκρουσης. Έπειτα διατρέχεται το αρχείο καταγραφής (από το αρχικό ή το τελευταίο σημείο ελέγχου) και προστίθενται στον πίνακα συναλλαγών, όλες οι συναλλαγές με τις εγγραφές Begin Transaction. Όταν εμφανιστεί μια εγγραφή End Log, η αντίστοιχη συναλλαγή αφαιρείται. Επιπλέον, ο

τελευταίος αριθμός ακολουθίας (Sequence Number) για κάθε συναλλαγή διατηρείται.

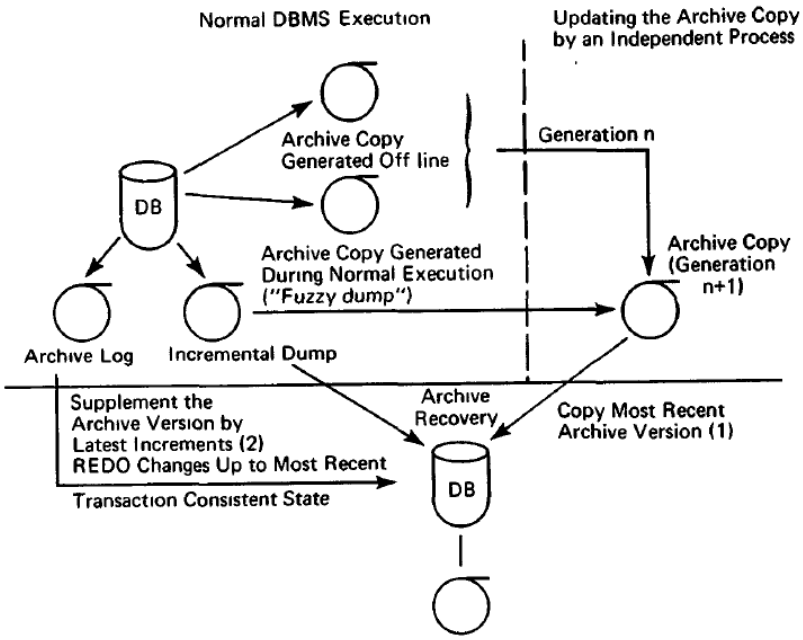
Παράλληλα με την προηγούμενη διαδικασία που αναφέρθηκε, ο πίνακας βρώμικων σελίδων συμπληρώνεται με την προσθήκη μιας νέας εγγραφής κάθε φορά που συναντάμε μια σελίδα που έχει τροποποιηθεί και δεν ανήκει στον πίνακα DPT. Αυτό όμως υπολογίζει μόνο ένα υπερσύνολο όλων των βρώμικων σελίδων κατά τη στιγμή της συντριβής, δεδομένου ότι δεν ελέγχεται το πραγματικό αρχείο βάσης δεδομένων ακόμα και αν η σελίδα γράφτηκε πίσω στον δίσκο.

4.4.3.2 Επανάληψη

Η διαδικασία ξεκινάει από τον πίνακα τροποποιημένων σελίδων DPT ώστε να μπορέσουμε να υπολογίσουμε τον ελάχιστο αριθμό ακολουθίας μιας τροποποιημένης σελίδας. Από εκείνο το σημείο πρέπει να αρχίσουμε να επαναλαμβάνουμε τις ενέργειες μέχρι την κατάρρευση, σε περίπτωση που δεν έχουν ήδη επαναληφθεί.

Κατά τη διάρκεια της διέλευσης από το αρχείο καταγραφής, ελέγχουμε για κάθε καταχώρηση, εάν υπάρχει η τροποποιημένη σελίδα P για την είσοδο στον πίνακα DPT. Στην περίπτωση που δεν υπάρχει, δεν είναι αναγκαία η επανάληψη καθώς τα δεδομένα παραμένουν στο δίσκο. Αν η σελίδα P υπάρχει στον πίνακα DPT, τότε θα παρατηρήσουμε αν ο αριθμός ακολουθίας του DPT είναι μικρότερος από τον αριθμό της εγγραφής ημερολογίου (δηλαδή αν η αλλαγή του ημερολογίου είναι πρόσφατη από την τελευταία έκδοση). Αν δεν είναι, τότε δεν θα ξανακάνει την είσοδο, δεδομένου ότι η αλλαγή είναι ήδη εκεί. Αν είναι, θα επιστραφεί η σελίδα από την βάση δεδομένων και θα ελεγχθεί ο αριθμός ακολουθίας που είναι αποθηκευμένος στην σελίδα με τον αριθμό ακολουθίας στο αρχείο καταγραφής. Αν ο πρώτος είναι μικρότερος από το δεύτερο, η σελίδα δεν χρειάζεται να γραφτεί στο

δίσκο. Ο έλεγχος είναι αναγκαίος, διότι η ανάκτηση DPT είναι μόνο ένα συνεχές υπερσύνολο των σελίδων που πραγματικά χρειάζονται αλλαγές για να μπορεί να ξαναχρησιμοποιηθεί. Τέλος, όταν όλοι οι προαναφερθέντες έλεγχοι ολοκληρωθούν και αποτύχουν, εφαρμόζεται εκ νέου η φάση της Επανάληψης και αποθηκεύεται ο νέος αριθμός ακολουθίας στη σελίδα. Είναι επίσης σημαντικό για την ανάκτηση από μια κατάρρευση κατά τη διάρκεια της Επανάληψης, καθώς η επανάληψη δεν εφαρμόζεται δύο φορές στην ίδια σελίδα.



Στην παραπάνω εικόνα φαίνεται ένα παράδειγμα της Επανάληψης.

4.4.3.3 Αναίρεση

Μετά τη φάση της Επανάληψης, η Βάση Δεδομένων παρουσιάζει την ακριβή κατάσταση της στιγμή της συντριβής. Ωστόσο, οι αλλαγές των μη επιβεβαιωμένων συναλλαγών είναι αναγκαίες για την επαναφορά της ΒΔ σε σταθερή κατάσταση.

Γι’ αυτό γίνεται μια διέλευση στο αρχείο καταγραφής προς τα πίσω για κάθε συναλλαγή του πίνακα TT (αυτές οι διαδρομές μπορούν φυσικά να συνδυαστούν σε μία) χρησιμοποιώντας στις εγγραφές τα πεδία της στήλης Προηγούμενος Αριθμός Ακολουθίας. Για κάθε εγγραφή αναιρούνται οι αλλαγές (χρησιμοποιώντας τις πληροφορίες στο πεδίο της Αναίρεσης) και εγγράφονται σε ένα αρχείο καταγραφής αποζημίωσης (Compensation log record) στο αρχείο καταγραφής. Αν εμφανιστεί μια συναλλαγή Begin Transaction τότε αποθηκεύεται μια εγγραφή End Log για την συγκεκριμένη συναλλαγή.

Οι εγγραφές αποζημίωσης καθιστούν δυνατή την ανάκτηση κατά τη διάρκεια μιας σύγκρουσης που συνέβη στη φάση της αποκατάστασης. Αυτό δεν είναι τόσο ασυνήθιστο όσο θα πίστευε κανείς, δεδομένου ότι είναι δυνατόν για την φάση της ανάκαμψης να πάρει αρκετά μεγάλο χρονικό διάστημα. Οι CLRs διαβάζονται κατά τη φάση της ανάλυσης και επαναλαμβάνονται κατά τη φάση της Επανάληψης.

Ακολουθεί ένα παράδειγμα εφαρμογής του αλγορίθμου. Έστω το παρακάτω αρχείο μετά από ένα συμβάν κατάρρευσης:

0	BEGIN CHECKPOINT
5	END CHECKPOINT (EMPTY XACT TABLE AND DPT)
10	T1: UPDATE P1 (OLD: YYY NEW: ZZZ)
15	T1: UPDATE P2 (OLD: WWW NEW: XXX)
20	T1: COMMIT

Εικόνα 23 - Παράδειγμα εφαρμογής του αλγορίθμου

Στη φάση της Ανάλυσης προκύπτουν τα εξής: αρχικά γίνεται μια σάρωση του αρχείου προς τα εμπρός ξεκινώντας από το LSN=0. Όταν το LSN=5 τότε αρχικοποιείται με μηδέν οι πίνακες XACT και DPT. Όταν το LSN=10 τότε προστίθενται στον πίνακα XACT, η T1 με την εντολή Add (T1, LSN 10) και αντίστοιχα στον πίνακα DPT η P1 με την εντολή Add (P1, LSN 10). Όταν το

LSN=15 τότε το πεδίο του πίνακα XACT τίθεται LastLSN=15 για την T1 και στον πίνακα DPT προστίθεται η P2 με την εντολή Add (P2, LSN 15). Τέλος όταν το LSN=20 τότε η κατάσταση της T1 τίθεται σε Commit στον πίνακα XACT.

Στη φάση της Επανάληψης ξεκινά μια σάρωση του αρχείου από το LSN=10. Όταν το LSN=10 διαβάζεται η σελίδα P1 και ελέγχεται ο αριθμός PageLSN που αποθηκεύεται στη σελίδα. Αν το PageLSN<10, επαναλαμβάνεται το LSN=10 και ορίζεται PageLSN=10. Όταν το LSN=15 διαβάζεται η σελίδα P2 και ελέγχεται ο αριθμός PageLSN που έχει αποθηκευθεί στη σελίδα. Αν ο αριθμός PageLSN<15, επαναλαμβάνεται το LSN=15 και τίθεται ο αριθμός της σελίδας σε PageLSN=15.

Στη φάση της Αναίρεσης δεν παρατηρείται τίποτα για να αναιρεθεί.

4.4.4 Σημεία ελέγχου

Για να αποφευχθεί ξανά η σάρωση όλου του αρχείου καταγραφής κατά τη διάρκεια της Ανάλυσης, συνιστάται η αποθήκευση των πινάκων DPT και TT στο αρχείο καταγραφής, σχηματίζοντας ένα σημείο ελέγχου. Από εκείνο το σημείο είναι πιθανή η επαναφορά των πινάκων DTT και TT καθώς τη στιγμή της κατάρρευσης βρίσκονται στην κατάσταση ανάγνωσής του. Η διαδικασία προχωρά στη συνέχεια με τη φάση της Επανάληψης και της Αναίρεσης (Mohan,2004).

Η δημιουργία σημείων ελέγχου περιλαμβάνει το κλείδωμα ολόκληρης της βάσης δεδομένων για να αποφευχθούν αλλαγές στους πίνακες DPT και TT. Παρατηρείται μια ασαφής καταγραφή κατά την αποθήκευση δύο εγγραφών. Η ασαφής εγγραφή ξεκινά και, μετά την προετοιμασία των δεδομένων του σημείου ελέγχου, το πραγματικό σημείο ελέγχου. Μεταξύ των δύο εγγραφών μπορούν να δημιουργηθούν άλλες εγγραφές αρχείων. Κατά τη διάρκεια της

ανάκαμψης, είναι αναγκαίο να βρεθούν οι δύο εγγραφές για τον προσδιορισμό του έγκυρου σημείου ελέγχου.

4.5 Μελλοντικές εκδόσεις του αλγορίθμου ARIES

Ο αλγόριθμος ARIES-RRH (Algorithm for Recovery and Isolation Exploiting Semantics with Restricted Repeating of History) (Mohan et al,1991) είναι μια βελτιωμένη έκδοση του αλγόριθμου επαναφοράς ARIES. Οι διορθώσεις στον ARIES-RRH αφορούν την ποσότητα των ενημερώσεων της Αναίρεσης που πρέπει να διενεργούνται από τη στιγμή της επανεκκίνησης του συστήματος, προκειμένου να έρθει η βάση δεδομένων σε σταθερή κατάσταση. Προσπαθεί να ελαχιστοποιήσει το μέγεθος επανάληψης του ιστορικού που πρέπει να εκτελεστεί.

4.5.1 ARIES-RRH

Ο ARIES-RRH απαιτεί ότι, για κάθε σελίδα, όλες οι ενημερώσεις καταγράφονται τουλάχιστον μέχρι το σημείο της πιο πρόσφατης διεκπεραιωμένης ή της ενημέρωσης για τη συγκεκριμένη σελίδα που απαιτείται να επαναληφθεί, εάν οι ενημερώσεις αυτές δεν υπάρχουν ήδη στη σελίδα. Το τελευταίο προσδιορίζεται συγκρίνοντας το LSN της σελίδας με τις LSN των σχετικών εγγραφών του αρχείου. Για τα δεδομένα για τα οποία χρησιμοποιείται η σελίδα, ο κανόνας αυτός συνεπάγεται ότι όλες οι «χαμένες» συναλλαγές καταγράφονται αλλά οι χαμένες ενημερώσεις χρειάζεται να επαναληφθούν, όπως συνέβη με τη DB2 V1 (Mohan, 1999). Αποδεικνύεται ότι η εφαρμογή του συγκεκριμένου κανόνα δεν είναι επαρκής, εφόσον κάποια συναλλαγή μπορεί να έχει ήδη επιστρέψει πίσω όταν έχει προκύψει κατάρρευση του συστήματος και ως αποτέλεσμα ορισμένα CLR θα έχουν γραφτεί, τα οποία επέζησαν από την αποτυχία του συστήματος. Μερικές σελίδες που επηρεάστηκαν από τις ενημερώσεις των CLRs μπορεί

να μην έχουν γραφεί στο δίσκο μετά εφόσον έχουν πραγματοποιηθεί αναιρέσεις.

4.5.2 ARIES/NT

Ο αλγόριθμος ARIES/NT (Algorithm for Recovery and Isolation Exploiting Semantics for Nested Transactions) (Mohan,1999) είναι μια επέκταση του αλγορίθμου ARIES, ο οποίος είχε αρχικά σχεδιαστεί για το μοντέλο πρώτου-επιπέδου συναλλαγής. Ο ARIES/NT εφαρμόζεται σε ένα πολύ γενικό μοντέλο των ένθετων συναλλαγών [HaRo87, HaRo93], που περιλαμβάνει τη μερική επιστροφή των υποσυναλλαγών, την ανοδική και καθοδική κληρονομικότητα των κλειδωμάτων, και ταυτόχρονη εκτέλεση των αρχικών και τελικών υποσυναλλαγών. Η αρχιτεκτονική του συστήματος περιλαμβάνει επίσης τις πτυχές της κατανεμημένης βάσης δεδομένων διαχείρισης.

Όπως και στον ARIES, στον ARIES/NT, η διαδικασία της επανεκκίνησης ξεκινά με την ανάλυση, συνεχίζει με την επανάληψη και ολοκληρώνεται με την αναίρεση. Η διαδικασία της Επανάληψης στον ARIES/NT λειτουργεί όμοια όπως και στον ARIES, ενώ οι αλγόριθμοι της Ανάλυσης και της Αναίρεσης έχουν τροποποιηθεί ώστε να υποστηρίξουν τα περιεχόμενα του αρχείου σε δενδρική δομή. Στον αλγόριθμο ARIES/NT το πεδίο UndoNxtLSN της CLR περιέχει ένα σύνολο καταγραφής διευθύνσεων και όχι ένα απλό LSN όπως στον αλγόριθμο ARIES.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Σκοπός της πτυχιακής εργασίας ήταν η βιβλιογραφική έρευνα των τεχνικών ανάκαμψης στα Συστήματα Διαχείρισης Βάσεων Δεδομένων. Η συναλλαγή αποτελεί την εργασία ώστε μια λειτουργία να διεκπεραιωθεί άμεσα και αποτελεσματικά. Συγκεκριμένα πρόκειται για μια αλλαγή που λαμβάνει χώρα στη Βάση Δεδομένων. Μια συναλλαγή υποστηρίζει την ακεραιότητα των δεδομένων της όταν ακολουθεί το μοντέλο ACID. Η ατομικότητα, η συνέπεια, η απομόνωση και η μονιμότητα αποτελούν ιδιότητες που πρέπει να καλύπτει μια συναλλαγή. Η εκτέλεση συναλλαγών που τρέχουν στο σύστημα ορίζει το χρονοπρόγραμμα το οποίο διακρίνεται στο σειριακό και στο σειριοποιήσιμο. Ωστόσο παρατηρούνται προβλήματα συγκρούσεων τα οποία προέρχονται και από τα δύο προαναφερθέντα χρονοπρογράμματα.

Με την εξέλιξη των ΣΔΒΔ η σειριακή εκτέλεση των συναλλαγών αντικαταστάθηκε σταδιακά από την εναλλασσόμενη εκτέλεση με αποτέλεσμα τα ΣΔΒΔ να αντιμετωπίζουν προβλήματα ως προς την άμεση και ασφαλή αποθήκευση των δεδομένων. Πιο αναλυτικά, παρουσιάστηκαν τρεις περιπτώσεις όπου ο συντονισμός των συναλλαγών είναι απαραίτητος για την επίλυση σημαντικών προβλημάτων που εμφανίζονται από την ταυτόχρονη εκτέλεση των συναλλαγών.

Μια τεχνική επίλυσης των συγκρούσεων σε χρονοπρογράμματα αποτελούν τα κλειδώματα ανάγνωσης και αποθήκευσης που χορηγούνται κάθε φορά σε μια συναλλαγή ώστε να αποφεύγονται προβλήματα. Το πρωτόκολλο 2PL ορίζει δυο βασικούς κανόνες για τη διαμοίραση κλειδιών και την παραλαβή αυτών μετά την εκτέλεση των συναλλαγών.

Το τελικό στάδιο της εργασίας αποτέλεσαν οι μέθοδοι ανάκαμψης του συστήματος ΒΔ από συγκρούσεις συναλλαγών και προτείνονται λύσεις σε ανάλογες περιπτώσεις. Επιπλέον, η εφαρμογή μιας συστάδας πρωτοκόλλων

με την ονομασία ARIES, σε ΣΔΒΔ που κατέρρευσαν από αποτυχίες του λογισμικού είτε του υλικού, οδήγησε στην δημιουργία νέων εκδόσεων του αλγορίθμου ώστε να επαναφέρεται το ΣΔΒΔ γρήγορα και να ανακτά τα δεδομένα από τα σημεία ελέγχου που καταγράφηκαν τη δεδομένη χρονική στιγμή. Ο ARIES-RRH απαιτεί ότι, για κάθε σελίδα, όλες οι ενημερώσεις καταγράφονται τουλάχιστον μέχρι το σημείο της πιο πρόσφατης διεκπεραιωμένης ή της ενημέρωσης για τη συγκεκριμένη σελίδα που απαιτείται να επαναληφθεί, εάν οι ενημερώσεις αυτές δεν υπάρχουν ήδη στη σελίδα. Από την άλλη πλευρά, ο ARIES/NT εφαρμόζεται σε ένα πολύ γενικό μοντέλο των ένθετων συναλλαγών που περιλαμβάνει τη μερική επιστροφή των υποσυναλλαγών, την ανοδική και καθοδική κληρονομικότητα των κλειδωμάτων, και ταυτόχρονη εκτέλεση των αρχικών και τελικών υποσυναλλαγών. Συγκριτικά και οι εκδόσεις κατά τη διαδικασία της επανεκκίνησης ξεκινούν με την ανάλυση, συνεχίζουν με την επανάληψη και ολοκληρώνονται με την αναίρεση.

Μια πρόταση προς διερεύνηση του συγκεκριμένου θέματος αποτελεί η εφαρμογή του αλγορίθμου ARIES σε κατανεμημένα συστήματα βάσεων δεδομένων καθώς και η καταγραφή των επιδόσεων από την επαναφορά των ΚΣΔΒΔ.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Bernstein, P. Hadzilacos, V. Goodman, N.(1987). Concurrency Control and Recovery in Database Systems. Addison Wesley Publishing Company. ISBN 0-201-10715-5.
2. Bernstein,P., Newcomer, E. (2009). Principles of Transaction Processing, 2nd Edition, Morgan Kaufmann (Elsevier), ISBN 978-1-55860-623-4
3. Gray, J. (1981). The Transaction Concept: Virtues and Limitations. Proceedings of the 7th International Conference on Very Large Databases. Cupertino, CA: Tandem Computers (pp. 144–154).
4. Gray, J., Reuter, A. (1993). Distributed Transaction Processing: Concepts and Techniques. Morgan Kaufmann. ISBN 1-55860-190-2.
5. Haerder, T., Rothermel, K. (1993). Concurrency Control Issues in Nested Transactions, VLDB Journal, Vol. 2, No. 1.
6. Mohan, C. (1992). ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging, ACM Transactions on Database Systems (pp. 94–162). New York, USA
7. Mohan, C. (1999). Repeating history beyond ARIES. Proceedings of the 25th International Conference on Very Large Data Base. Morgan Kaufmann (pp 1-17). Edinburgh, Scotland
8. Mohan, C. (2004). ARIES family of locking and recovery algorithms http://www.almaden.ibm.com/u/mohan/ARIES Impact.html. Τελευταία πρόσβαση στον ιστότοπο στις 12/10/2015
9. Padua, D. (2011). Encyclopedia of Parallel Computing. Springer (p. 524). ISBN 9780387097657.

10. Raz, Y. (1992). The Principle of Commitment Ordering, or Guaranteeing Serializability in a Heterogeneous Environment of Multiple Autonomous Resource Managers Using Atomic Commitment. Proceedings of the Eighteenth International Conference on Very Large Data Bases (VLDB) (pp. 292-312). Vancouver, Canada.
11. Weikum, G. Vossen, G. (2001). Transactional Information Systems, Elsevier, 2001, ISBN 1-55860-508-8
12. Μανωλόπουλος, Ι. Παπαδόπουλος, Α. (2003). Συστήματα Βάσεων Δεδομένων, Θεωρία και Πράξη. Παρατηρητής. Θεσσαλονίκη, Ελλάδα

ΠΗΓΕΣ ΑΠΟ ΤΟ ΔΙΑΔΙΚΤΥΟ

- http://web.stanford.edu/class/cs340v/papers/recovery.pdf
- http://www.tutorialspoint.com/dbms/dbms_data_recovery.htm
- https://www.usenix.org/system/files/conference/osdi14/osdi14-paper-zheng_wenting.pdf
- http://www.adms-conf.org/kimura_adms12.pdf