

טרנזקציות ועבודה במקביל

לרוב, העבודה מול בסיס הנתונים לא תבצע באופן ישיר, אלא באמצעות תוכנה. בהרבה מקרים, תהליכים שמבוצעים בתוכנה כוללים בתוכם מספר פניות לבסיס הנתונים, כאשר כל פנייה שולפת, מעדכנת, מוספיה או מוחקת נתונים שונים בבסיס הנתונים, וזאת על מנת להשלים את אותו התהליך. המונח טרנזקציה (transaction) מתייחס לאוסף הפקודות SQL (לעדכון בסיס הנתונים), שמתבצעות כיחידת עבודה לוגית בודדת, שיכולה להסתיים בהצלחה או בכישלון (במקרה זה כל פעולות העדכון שהתרחשו יבוטלו). מטרת העל של הטרנזקציה הינה שמירה על אמינות ושלמות בסיס הנתונים בסביבה עתירת תנועות ומרובת משתמשים. בסיס נתונים צריך להבטיח את הצלחת רצף פקודות העדכון ולא רק את הצלחת הפקודה הבודדת (בשיטת "הכל או כלום") שכן לא ניתן לבצע טרנזקציה באופן חלקי.

דוגמא: הפקדת צ'ק - הפקדת צ'ק של לקוח מהווה פעולה בודדת עבור הלקוח, אך בפועל מתבצעות שתי פעולות עדכון, (1) החסרת סכום הצ'ק מחשבון א', (2) הוספת סכום הצ'ק לחשבון ב'.

על מנת לשמור על שלמות הנתונים, נדרוש שבסיס הנתונים יאפשר את עקרונות היסוד הבאים בהקשר של טרנזקציות: (1) אטומיות (Atomicity) (2) עקביות (Consistency) (3) בידוד (Isolation) (4) עמידות (Durability). ארבעת עקרונות אלו ידועים בשם The ACID properties. אטומיות - (Atomicity) הכל או לא כלום. או שכל פעולות הטרנזקציה יצאו לפועל בשלמותם או במקרה של כישלון, החלק שבוצע יבוטל (rollback) ואז שום פעולה לא יצאה לפועל. כישלון לא יאפשר להשאיר את מסד הנתונים במצב שטרנזקציה בוצעה באופן חלקי בלבד.

עקביות - (Consistency) תנועה חייבת להעביר את בסיס הנתונים ממצב תקין אחד למצב תקין אחר אפילו שתוך כדי פעולתה התנועה מפירה זמנית את תקינות בסיס הנתונים.

בידוד - (Isolation) מערכת עם מספר טרנזקציות המבוצעות במקביל, צריכה לספק מכניזם לבידוד טרנזקציות מהשפעתם של טרנזקציות אחרות הרצות במקביל, זאת אומרת שתנועות חייבות להתבצע באופן בלתי תלוי זו בזו.

עמידות (Durability) - לאחר שטרנזקציה הסתיימה בהצלחה, השינויים שהיא ביצעה למסד הנתונים ישמרו במסד הנתונים לתמיד ולא יאבדו בעתיד בשום תרחיש.

בפועל, על מנת לשמר את עיקרון האטומיות, בסיס הנתונים עוקב אחר הערכים המקוריים של המשתנים עבורם מתבצעת פעולת Write, ואם הטרנזקציה לא מסתיימת בהצלחה, בסיס הנתונים משחזר את הערכים המקוריים כך שבמסד הנתונים תתקבל חזרה התמונה המקורית, כאילו שהטרנזקציה כלל לא פעלה (ע"י ביצוע גלגול לאחור).

על מנת להתחיל טרנזקציה חדשה נרשום BEGIN TRANSACTION ולאחריה יופיע סט של פקודות. אם הפקודות הסתיימו בהצלחה, נסיים את הטרנזקציה עם פקודת COMMIT (כל הפקודות תחת הטרנזקציה יבוצעו בפועל והשינויים שבוצעו בעקבותיהם יישמרו בבסיס הנתונים באופן תמידי וסופי), אחרת נבצע פקודת ROLLBACK (כל הפקודות תחת הטרנזקציה ייכשלו ויבוצע ביטול של כל הפקודות הנ"ל).

המנגנון שמאפשר להפעיל את פקודת ה-ROLLBACK ולחזור למצב המקורי הינו יומן האירועים (Log File), קובץ המנוהל ע"י בסיס הנתונים ומכיל רשומה עבור כל פקודת עדכון שבוצעה, וכך הוא מאפשר התאוששות ושחזור של המצב המקורי לפי פעולות העדכון. במקרה של ROLLBACK, יומן האירועים יקרא בסדר כרונולוגי הפוך, זאת אומרת מהפקודה האחרונה שרשומה לפני פקודת הגלגול האחרון (הפקודה האחרונה שבוצעה) ועד לפקודה הראשונה בתנועה, במטרה להחזיר את בסיס הנתונים למצבו שלפני העדכון.

ראינו שאחת התכונות החשובות של טרנזקציות היא בידוד (isolation). כאשר מספר טרנזקציות מורצות במקביל יתכן ותכונת הבידוד לא תשמר. בכדי לשמור על תכונה זו, המערכת חייבת לבצע בקרה של האינטראקציה בין הטרנזקציות הרצות במקביל, בקרה זו מושגת ע"י פרוטוקול לבקרת בו-זמניות. קיימים מספר פרוטוקולים לתמיכה בבקרת בו זמניות בהם נוכל להשתמש כאשר נריץ מספר טרנזקציות במקביל.

השיטה הנפוצה ביותר למימוש דרישה זו היא לאפשר לטרנזקציה לגשת לפריט מידע רק כאשר הטרנזקציה ביצעה נעילה (lock) על פריט מידע זה. קיימות מספר אפשרויות. (1) נעילה משותפת - תוכנית המבקשת לקרוא רשומה מסוימת, תחזיק אותה בסטטוס "נעילה משותפת" ובכך תאפשר לתוכניות אחרות לקרוא במקביל את אותה הרשומה (אך לא תאפשר לעדכן אותה). ברגע שהתוכנית מבקשת לעדכן את השורה היא צריכה להעביר את הנעילה מ- "שיתופית" ל- "בלעדית". (2) - אף תוכנית יישום אחרת (טרנזקציה) אינה מורשית לקרוא או לעדכן את הנתונים בזמן שהטרנזקציה רצה. כאשר טרנזקציה רוצה להפעיל נעילה בלעדית על פריט מידע הנעול בנעילה בלעדית ע"י טרנזקציה אחרת, יהיה עליה להמתין עד לשחרור הפריט.

במצב של מבוי סתום (Dead Lock, נעילה ללא מוצא) אנו נמצאים בתרחיש שבו תוכנית יישום (אחת או יותר) נועלת אובייקט שהתוכנית השנייה מבקשת להשתמש בו. לדוגמא: טרנזקציה A מתחילה לפעול ושולפת שורה X ומבצעת לה נעילה בלעדית לקראת ביצוע פעולת עדכון. טרנזקציה B מתחילה לפעול ושולפת שורה Y ומבצעת לה נעילה בלעדית. טרנזקציה A רוצה לקרוא את שורה Y ולכן צריכה להמתין. טרנזקציה B רוצה לקרוא את שורה X ולכן צריכה להמתין. שתי הטרנזקציות "נעולות" ולכן לא יוכלו להמשיך.

נניח שיש מספר טרנזקציות שרצות במקביל, וחלק מהטרנזקציות מבקשות לבצע נעילה משותפת (נעילת קריאה) על פריט מידע לתקופת זמן קצרה מאוד. במקביל קיימת טרנזקציה שמעוניינת לקבל אישור לנעילה בלעדית על אותו פריט מידע. לנעילה משותפת יש קדימות, ולכן, כל עוד קיימות טרנזקציות אחרות שמסתפקות בביצוע נעילה משותפת, הטרנזקציה שמחכה לקבלת אישור לנעילה בלעדית, לא תקבל זאת ותמשיך להמתין. מצב מסוג זה נקרא מצב של "הרעבה".

בכדי למנוע הרעבות נשאף שאישור נעילה יינתן אם שני התנאים הבאים מתקיימים: (1) פריט המידע אינו נעול, או שניתן לבצע נעילה משותפת (2) אין טרנזקציה אחרת הממתנה לנעילת הפריט (הגישה בקשה קודמת).

השיטה הנפוצה ביותר לקביעת סדר ההרצה של הטרנזקציות (בצורה סדרתית) היא פרוטוקול תגי הזמן (Timestamp protocol). כל טרנזקציה Ti מקבלת טרם ריצתה חותמת זמן (timestamp) המסומנת ע"י TS(Ti), שהיא ייחודית וקבועה על ידי המערכת. חותמת הזמן תקבע בסדר עולה עבור כל טרנזקציה חדשה המתווספת לתזמון. תג הזמן של הטרנזקציות קובע את סדר הריצה.

במקרים שרוב הטרנזקציות בתזמון הינן טרנזקציות לקריאה בלבד, יחס ההתנגשויות בין הטרנזקציות יהיה נמוך יחסית, ולכן חלק ניכר מהטרנזקציות הללו גם אם הן יורצו ללא בקרת מקביליות ריצתם תסתיימנה במצב עקבי. בקרת המקביליות כופה תקורת תפעול נוספת ועלולה לגרום עיכובים במהלך הריצה ולכן לפעמים נעדיף להשתמש באלטרנטיבה בעלת תקורה נמוכה יותר. הבעיה היא שאנו לא יודעים מראש איזה טרנזקציות יהיו חלק מההתנגשויות, ולכן נצטרך סכמת ניתור שתוגדר בהמשך ותאפשר את הקטנת התקורה. אנו נניח שכל טרנזקציה Ti מפעילה מספר שלבים שונים במהלך חייה. Read Phase - במהלך שלב זה, המערכת מריצה את טרנזקציה וקוראת את הערכים של פרטי המידע ומאחסנת אותם במשתנים מקומיים זמניים (מבלי לעדכן עדיין את מסד הנתונים). Validation Phase - הטרנזקציה מבצעת בדיקה לקביעה האם היא יכולה להעתיק למסד הנתונים את המשתנים הזמניים מבלי לגרום לפגיעה בסדרתיות. Write Phase - אם הטרנזקציה מצליחה בבדיקת התקינות של השלב הקודם אזי המערכת מבצעת בפועל את העדכונים למסד הנתונים, אחרת מבוצע גלגול לאחור.