ניהול תנועות
חלק 3

Transaction Management
Part 3

בקרת מקביליות ללא נעילות

הקדמה

בקרת מקביליות ללא נעילות

- העסקאות מתבצעות ללא הגבלה
  - לא לגמרי מדויק – פרטים בהמשך
- יש צורך בבדיקות כדי לוודא שלא נוצר תזמון שאינו בר סדרתיות
- אם עיסקה אינה עוברת את הבדיקות, גורמים לה להתבטל (abort) ומתחילים אותה מחדש

הרעיון הבסיסי

- מחליטים מראש על תזמון סדרתי
- בודקים שהפעולות שעיסקה מבצעת שומרות על שקילות עם התזמון הסדרתי שעליו הוחלט
- משתמשים בחותמות זמן
  - כדי להחליט מהו התזמון הסדרתי
  - וכדי לבצע הבדיקות

אם עיסקה אינה עוברת את הבדיקות, צריך לבטל אותה

- כל עיסקה נבדקת
  - אם הבדיקות עוברות בהצלחה, העיסקה רשאית להמשיך ולבסוף גם להתחייב
  - אם עיסקה אינה עוברת את אחת הבדיקות, המשמעות היא שהעיסקה עלולה לגרום לתזמון שאינו בר סדרתיות, ולכן צריך לבטל אותה ולהתחיל להריץ אותה מחדש (עם חותמת זמן חדשה)

    ←מדוע נותנים חותמת זמן חדשה בכל פעם שהעיסקה מתחילה לרוץ?

ארבע ואריאציות

- בקרת מקביליות ע"י אימות
Concurrency Control by Validation
- בקרת מקביליות אופטימית
Optimistic Concurrency Control
- בקרת מקביליות ע"י חותמות זמן
Timestamp-Based Concurrency Control
- בקרת מקביליות מרובת גרסאות ע"י חותמות זמן
Multiversion Timestamps
- כולן למעט השנייה מתוארות בפרקים 18.8 ו- 18.9 של הספר
DB Systems the Complete Book by Garcia-Molina et al.
- השנייה מתוארת ב-
DB Management System by Ramakrishnan et al.

- בכולן חותמות זמן לעסקאות
- בשתיים האחרונות יש חותמות זמן גם לפריטים

בקרת מקביליות ע"י אימות

Concurrency Control by Validation

יהיה בסדר

- הרעיון הבסיסי הוא לדחות את הבדיקה עד לאחר שהעיסקה מסיימת לבצע את כל החישובים, מתוך הנחה שיש רק סיכוי קטן שיהיה צורך לבטל את העיסקה

שלושה שלבים של ביצוע עיסקה

- שלב הקריאה
  - לכל עיסקה מוקצה שטח עבודה פרטי בזיכרון של המחשב, שלתוכו קוראת העיסקה את הנתונים מהמסד
  - עדיף שטח עבודה בזיכרון הפנימי אם זה אפשרי
  - העיסקה מבצעת את כל החישובים והכתיבות בשטח העבודה הפרטי
- שלב הבדיקה (אימות)
- שלב הכתיבה
  - אם הבדיקה עברה בהצלחה, העיסקה מעתיקה את התוצאות הסופיות משטח העבודה הפרטי למסד

מניחים שאין שתי חותמות זמן זהות

המידע ששומרים עבור כל עיסקה

- START(T) הזמן שבו מתחילה העיסקה T
- VAL(T) הזמן שבו מתבצעת הבדיקה עבור T
- FIN(T) הזמן שבו מסיימת העיסקה T
- RS(T) היא קבוצת הפריטים ש- T קראה
- WS(T) היא קבוצת הפריטים ש- T כתבה
- אין צורך לשמור מידע זה לעד (בגלל הנאמר בשקפים 14 ו- 16)
  - עיסקה T הופכת לבלתי רלבנטית אם היא סיימה וזמן הסיום שלה (FIN(T קודם ל- (קרי, קטן מ-) זמני ההתחלה של כל העסקאות שכרגע עדיין רצות במערכת

הרעיון העיקרי של בקרה ע"י אימות

- התזמון שנוצר בפועל הינו שקול קונפליקטים לתזמון סדרתי שבו העסקאות מבוצעות לפי הסדר של זמני הבדיקה (האימות)
- במילים אחרות, אם עיסקה עוברת את שלב הבדיקה, אז אפשר לראות בשלב זה את הרגע שבו מתבצעת העיסקה כולה
- הדבר מחייב ששלב הבדיקה יתבצע בצורה אטומית
  - שלב זה מתבצע ללא מקביליות, כלומר העסקאות נבדקות זו אחר זו, לפי סדר הגעתן לשלב זה

מה צריך לבדוק?

- אסור שלעיסקה הנבדקת יהיו קונפליקטים שסותרים את השקילות עם התזמון הסדרתי שקבענו
  - כלומר, התזמון הסדרתי לפי זמני הבדיקות
- אין מספיק מידע כדי לבדוק במדויק שאין קונפליקטים אסורים (קרי, שסותרים השקילות)
- לכן בודקים שאין בכלל קונפליקטים שעלולים לגרום לבעיה
- צריך לבדוק עיסקה רק מול עסקאות שכבר עברו בהצלחה את שלב הבדיקה

מדוע בודקים רק מול עסקאות שכבר עברו בהצלחה את הבדיקה

- כי התזמון שנוצר בפועל, עד לרגע זה, כולל רק את העסקאות שכבר עברו בהצלחה את הבדיקה
- אם העסקה הנבדקת עוברת בהצלחה את הבדיקה, אז היא תצטרף לתזמון זה; אחרת, תבוטל
- בצורה זו נוצר תזמון שהוא שקול קונפליקטים לתזמון הסדרתי לפי חותמות הזמן VAL(T)

בודקים שאין קונפליקט קריאה-כתיבה

- T נבדקת כרגע ו- U כבר עברה בהצלחה את הבדיקה
- בודקים שאין אף פריט ש- T קראה וש- U כתבה
- קרי, בודקים ש- $WS(U) \cap RS(T) = \varnothing$ לכל עיסקה U שעברה בהצלחה את הבדיקה ו- $START(T) < FIN(U)$
  - אם U טרם סיימה, אז ברור ש- $START(T) < FIN(U)$ מתקיים
- בדיקה זו מבטיחה שאין בכלל קונפליקט קריאה-כתיבה בין T לבין U
- אם $START(T) < FIN(U)$ (כלומר, U סיימה לפני ש- T התחילה), אז ברור ש- T קוראת ערכים רק אחרי ש- U כתבה אותם ולכן לא יכול להיות ביניהן קונפליקט שסותר את השקילות לתזמון הסדרתי

אין צורך לבדוק קונפליקט כתיבה-קריאה

- בין עיסקה T שכרגע נבדקת, לבין עיסקה U שעברה הבדיקה בהצלחה לא יכול להיות קונפליקט כתיבה-קריאה שסותר השקילות, כי
  - T כותבת רק אחרי שהיא עוברת בהצלחה את שלב הבדיקה
  - U סיימה לקרוא מהמסד לפני שנבדקה
  - לכן, לא יתכן שעיסקה U שכבר נבדקה קראה ערך שנכתב ע"י עיסקה T שרק עכשיו נבדקת

בודקים שאין קונפליקט כתיבה-כתיבה

- T נבדקת כרגע ו- U כבר עברה בהצלחה את הבדיקה
- בודקים שאין אף פריט שגם T וגם U כותבות
- קרי, בודקים ש- $WS(U) \cap WS(T) = \emptyset$ לכל עיסקה U שעברה בהצלחה את הבדיקה ו- $FIN(U) > VAL(T)$
  - אם U טרם סיימה, אז ברור ש- $FIN(U) > VAL(T)$ מתקיים
- בדיקה זו מבטיחה שאין בכלל קונפליקט כתיבה-כתיבה בין T לבין U
- אם $FIN(U) > VAL(T)$ (כלומר, U סיימה לפני ש- T נכנסה לשלב הבדיקה), אז ברור ש- T כותבת ערכים רק אחרי ש- U כתבה אותם (אם בכלל) ולכן לא יכול להיות ביניהן קונפליקט כתיבה-כתיבה שסותר את השקילות לתזמון הסדרתי

סיכום:
הבדיקות בבקרת מקביליות ע"י אימות

- כשעיסקה T נמצאת בשלב הבדיקה, מוודאים ש-
  - $WS(U) \cap RS(T) = \varnothing$ לכל עיסקה U שכבר עברה בהצלחה את הבדיקה ו- $FIN(U) > START(T)$
  - $WS(U) \cap WS(T) = \varnothing$ לכל עיסקה U שכבר עברה בהצלחה את הבדיקה ו- $FIN(U) > VAL(T)$

דוגמה 18.29

START(T)
START(U)
VAL(U)
VAL(T)
START(V)
FIN(U)
START(W)
VAL(V)
FIN(T)
VAL(W)fails
FIN(V)

- RS(U) = {B} WS(U) = {D}
- RS(T) = {A,B} WS(T) = {A,C}
- RS(V) = {B} WS(V) = {D,E}
- RS(W) = {A,D} WS(W) = {A,C}

- בודקים ש- $RS(T) \cap WS(U) = \varnothing$ לכל עיסקה U שעברה בהצלחה את הבדיקה ו- $START(T) < FIN(U)$
- בודקים ש- $WS(T) \cap WS(U) = \varnothing$ לכל עיסקה U שעברה בהצלחה את הבדיקה ו- $VAL(T) < FIN(U)$

בקרת מקביליות אופטימית

Optimistic Concurrency Control

בקרת מקביליות אופטימית
Optimistic Concurrency Control

- בהשוואה לבקרת מקביליות ע"י אימות, בגישה האופטימית מרשים קונפליקטים של כתיבה-כתיבה, אבל דואגים שהם יהיו בסדר הנכון (כלומר, יתאימו לתזמון הסדרתי שקבענו)

השלבים העיקריים של ביצוע עיסקה בגישה האופטימית – כמו קודם

- שלב הקריאה
  - לכל עיסקה מוקצה שטח עבודה פרטי בזיכרון של המחשב, שלתוכו קוראת העיסקה את הנתונים מהמסד
  - עדיף שטח עבודה בזיכרון הפנימי אם זה אפשרי
  - העיסקה מבצעת את כל החישובים והכתיבות בשטח העבודה הפרטי
- שלב הבדיקה
- שלב הכתיבה
  - אם הבדיקה עברה בהצלחה, העיסקה מעתיקה את התוצאות הסופיות משטח העבודה הפרטי למסד

בגישה ע"י אימות, רק הבדיקה הייתה קטע קריטי

קטע קריטי

- הקטע שבו עיסקה מבצעת את שני השלבים האחרונים (בדיקה וכתיבה) הוא קטע קריטי
  - בכל נקודת זמן, המערכת מאפשרת רק לעיסקה אחת (לכל היותר) לבצע את השלבים הללו
  - וכל עיסקה מבצעת את שני השלבים הללו ברצף
- כלומר, העסקאות מבצעות את השלבים של בדיקה וכתיבה ברצף ובצורה סדרתית לחלוטין
  - בפרט, רק עיסקה אחת לכל היותר נבדקת או כותבת למסד הנתונים בכל רגע נתון

אז איפה המקביליות?

- הבדיקה לוקחת מעט זמן באופן יחסי לזמן הריצה של העיסקה
- גם הכתיבה לוקחת מעט זמן, כי כל החישובים כבר בוצעו ורק נשאר להעתיק את התוצאות הסופיות משטח העבודה הפרטי למסד
- לפיכך, רוב הזמן העיסקה מתבצעת במקביל לעסקאות אחרות

חותמת זמן (קודם קראנו לזה $VAL(T_i)$)

- כל עיסקה מקבלת חותמת זמן בתחילת שלב הבדיקה
- $TS(T_i)$ מסמן את חותמת הזמן של העיסקה T_i
- הרעיון העיקרי כמו קודם: התזמון הנוצר על ידי העסקאות שעוברות את הבדיקות הינו שקול קונפליקטים לתזמון הסדרתי שבו העסקאות מבוצעות לפי הסדר של חותמות הזמן
  - כלומר, עיסקה שעוברת את הבדיקה "כאילו" מתבצעת בצורה אטומית ברגע של תחילת הבדיקה

צריך לבדוק רק קריאה-כתיבה

- העובדה שהקטע הקריטי מורכב משני השלבים האחרונים (של הבדיקה והכתיבה) מונעת קונפליקטים של כתיבה-כתיבה שאינם תואמים את התזמון הסדרתי לפי חותמות הזמן
- הבדיקה שאין קונפליקטים אסורים של קריאה-כתיבה היא כמו בבקרת מקביליות ע"י אימות

סיכום:
הבדיקה בבקרת מקביליות אופטימית

- בשלב הבדיקה של עיסקה T_k מוודאים שמתקיים התנאי הבא:
- $WS(T_i) \cap RS(T_k) = \varnothing$ לכל עיסקה T_i שכבר עברה בהצלחה את הבדיקה ו- $START(T_k) < FIN(T_i)$

שיפור לגישה האופטימית

מה אפשר לשפר?

- כאשר עיסקה T_i עוברת בהצלחה את שלב הבדיקה, אפשר גם לבדוק האם יש עיסקה T_k שטרם החלה את שלב הבדיקה, אבל כבר קראה פריט A ש- T_i עומדת לכתוב
- אם יש עיסקה T_k כנ"ל, אז ברור שהיא תיכשל בשלב הבדיקה
- אפשר להרוג את T_k מייד (kill policy) או לתת לה למות כשתגיע לשלב הבדיקה (die policy)

המידע שצריך לשמור

- צריך טבלה עם רשומות מהצורה
  (transaction id, item id, modified)
- כל עיסקה, שקוראת פריט, מוסיפה רשומה כנ"ל לטבלה עם ה- id של העיסקה וה- id של הפריט
  - השדה modified הוא בהתחלה false
  - דוגמה: $(T_k,A,false)$
- כאשר עיסקה מסתיימת בהצלחה או מתבטלת, כל הרשומות שלה נמחקות מהטבלה

בשלב הכתיבה

- כאשר עיסקה T_i כותבת למסד פריט A, היא משנה *בכל* הרשומות עבור הפריט A את השדה modified ל- true
  - אם T_i כותבת, אז היא עברה הבדיקה בהצלחה
- אפשר להרוג מייד (kill policy) את כל העסקאות (חוץ כמובן מ- T_i עצמה), שאת הרשומות שלהן T_i שינתה, או
- לתת להן למות כשתגענה לשלב הבדיקה (die policy), כפי שמוסבר בשקף הבא

בשלב הבדיקה

- כאשר עיסקה T_k נבדקת, צריך לשלוף מהטבלה את כל הרשומות עבור עיסקה זו
- אם (לפחות) באחת הרשומות, עבור פריט כלשהו A, השדה modified הוא true, פירוש הדבר שעיסקה שכבר התחייבה כתבה ערך עבור A אחרי ש- T_k קראה את הערך של A
- לכן צריך, לבטל את T_k

הפעולות על הטבלה
מחייבות מנעול

- כאשר עיסקה מוסיפה רשומה לטבלה או משנה את השדה modified בכל הרשומות עבור פריט A, העיסקה צריכה מנעול בלעדי על הטבלה
- פירוט בשני השקפים הבאים

עסקה שקוראת נועלת הטבלה

- עיסקה נועלת את הטבלה לפני פעולת קריאה, ומשחררת את המנעול מיד לאחר שקראה והוסיפה את השורה המתאימה לטבלה, לדוגמה: $(T_k,A,false)$
- בכל רגע נתון, רק עיסקה אחת יכולה להחזיק מנעול על הטבלה

צריך לנעול גם
בשלבי הבדיקה והכתיבה

- עיסקה חייבת לנעול את הטבלה גם בשלבי הבדיקה והכתיבה
  - כי היא משנה את הטבלה לפני כתיבה
  - בנוסף, הנעילה בשני שלבים אלה מבטיחה שהם מתבצעים ברצף ובאופן סדרתי (קרי, רק עיסקה אחת בכל פעם)
- לפיכך, בזמן שעיסקה מבצעת שלבים אלה, אף עיסקה אחרת אינה יכולה לקרוא מהמסד (וכמובן שגם לא לכתוב למסד)
- ובזמן שעיסקה קוראת, אף עיסקה אחרת אינה יכולה להתחיל את שלבי הבדיקה והכתיבה

איזה גישה עדיפה?
להרוג מייד או לתת למות בבדיקה?

- לכאורה, עדיף להרוג מיד
- אבל אם מאפשרים לעיסקה להמשיך לרוץ עד שלב הבדיקה, היא תביא לזיכרון הפנימי (קרי, לחוצץ) את כל הבלוקים של המסד שהיא צריכה
- לכן, הריצה הבאה של העיסקה תהיה הרבה יותר קצרה ולפיכך יש סיכוי גדול יותר שהיא תעבור את שלב הבדיקה בהצלחה

כמובן, שבריצה הבאה,
כל הבלוקים של המסד שהעיסקה צריכה
כבר יהיו בזיכרון הפנימי בתנאי ש-

- יש מספיק מקום בזיכרון לשמור את כל הבלוקים (יחד עם בלוקים של עסקאות אחרות שרצות בינתיים)
- הבלוקים שהעיסקה צריכה אינם משתנים מריצה אחת לשניה
- <u>הערה</u>: אם העיסקה מתבטלת שוב, אז צריך להרוג אותה מיד, כי ממילא כל הבלוקים שלה כבר בזיכרון הפנימי