使用 DB2 9.5 的樂觀鎖定功能來提高並行性

新的樂觀鎖定功能可以避免長時間的鎖定

級別：中等

IBM，IBM 認證的 IT 專家 Werner Schuetz (werner_schuetz@de.ibm.com)

2008 年 1 月 17 日

IBM® DB2®, Version 9.5 for Linux®, UNIX®, and Windows® 提供加強型樂觀鎖定支援，這是一種 SQL 資料庫應用技術，在選取和更新或刪除資料列期間，不會讓資料列保持在鎖定狀態。本文將會說明這項加強功能，並讓您瞭解使用此程式設計模型的應用程式會如何因這項加強的樂觀鎖定功能而獲益，從而提高並行性。

前言

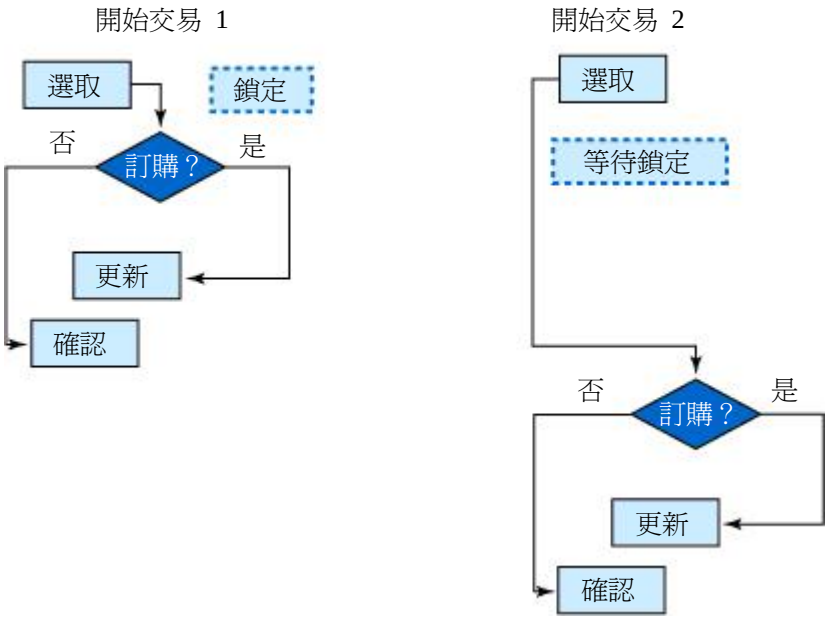
悲觀鎖定及樂觀鎖定

悲觀鎖定 (Pessimistic locking)

悲觀鎖定策略假定其他使用者會嘗試修改您正在變更表格中的同一個資料列的可能性很高。因此，在選取一個資料列後到嘗試對該資料列進行搜尋的更新或刪除作業期間會保持鎖定狀態（例如，使用可重複讀取隔離層次或將該表格鎖定在專用模式）。悲觀鎖定的優點是可確保變更的一致性和安全性，主要缺點則是這種鎖定策略可能不太具有延展性。在有許多使用者或長期交易的系統上，或者當交易涉及大量的實體時，必須等待鎖定解除的可能性就會增加。

圖 1 說明悲觀鎖定的運作方式。交易 1 讀取特定的記錄，並將該資料列鎖定，它花了一些時間決定是否要更新該資料列。在此期間，交易 2 想要存取同一個資料列，但它必須等待，直到交易 1 解除鎖定爲止。屆時，交易 2 將會收到 SELECT 傳回的結果，然後便可以繼續進行其商業邏輯。

圖 1. 悲觀鎖定的概念

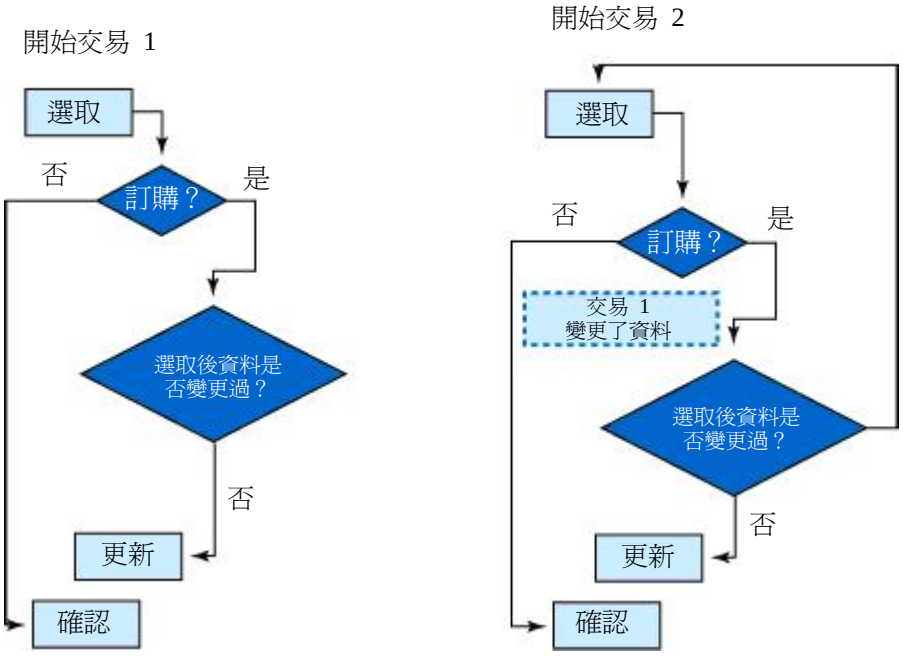


樂觀鎖定 (Optimistic locking)

悲觀鎖定方法的主要問題是交易必須彼此等待。避免這個問題的方法之一就是採用樂觀鎖定策略，並假定其他使用者不太可能會嘗試變更您正在變更的同一個資料列。如果該資料列真的變更了，則更新或刪除作業將會失敗，而應用程式邏輯會透過如重試選取等方式來處理這類失敗作業。使用這個方法，就不會在選取和更新或刪除資料列期間保持鎖定狀態。但是，這個方法需要在讀取和變更資料期間，確保該資料並未變更。雖然應用程式需要更多的重試邏輯，但是樂觀鎖定策略的主要優點是，可以讓其他交易無法使用特定資源的時間縮到最短，因此和悲觀鎖定比起來算是較具延展性的鎖定替代方案。

圖 2 說明樂觀鎖定背後的觀念。和 圖 1 類似，交易 1 讀取特定的記錄，但接著便解除其鎖定，現在，交易 2 便可擷取同一個資料列。在確定交易之前，交易 1 和 交易 2 都必須檢查前次執行 SELECT 後，該資料列是否已變更。如果已經變更，該交易必須以新的 SELECT 重新啓動，才能擷取現行資料。但是，如果該資料列在前次 SELECT 後並未經過變更，就可以順利更新資料。

圖 2. 樂觀鎖定的概念



DB2 9.5 加強的樂觀鎖定功能

DB2 9.5 中的樂觀鎖定功能讓其他交易無法使用特定資源的時間縮到最短，因此提高了延展性。因爲資料庫管理程式可以確定變更資料列的時間，因此可以確保資料的完整性，同時限制保持鎖定的時間。使用樂觀並行控制功能，資料庫管理程式便可以在讀取作業完成後，立即解除資料列或頁面的鎖定。

DB2 9.5 for Linux, Unix, and Windows 新增的支援可輕鬆快速地進行樂觀鎖定，而不會發生正向誤判 (false positive) 的情形。您可使用下列新的 SQL 函數、表示式及功能來新增這項支援：

- **資料列識別碼（RID_BIT 或 RID）內建函數：**這個內建函數可用於 SELECT 清單或述詞陳述式中。在述詞中（例如 WHERE RID_BIT(tab)=?），實作 RID_BIT 等於述詞來作爲新的直接存取方法，以便有效地找到該資料列。之前所謂的用數值來樂觀鎖定數值是將所有選取的直欄值新增至述詞，再依賴某些獨特的直欄組合來僅限定單一的資料列，透過這種效率較低的存取方法來完成。

- **ROW CHANGE TOKEN 表示式：**這個新的表示式會傳回一個記號作爲 BIGINT，該記號代表資料列修改順序中的一個相對點。應用程式可以比較一個資料列的現行資料列變更記號值和前次提取該資料列時所

儲存的資料列變更記號值，以判斷該資料列是否已變更。

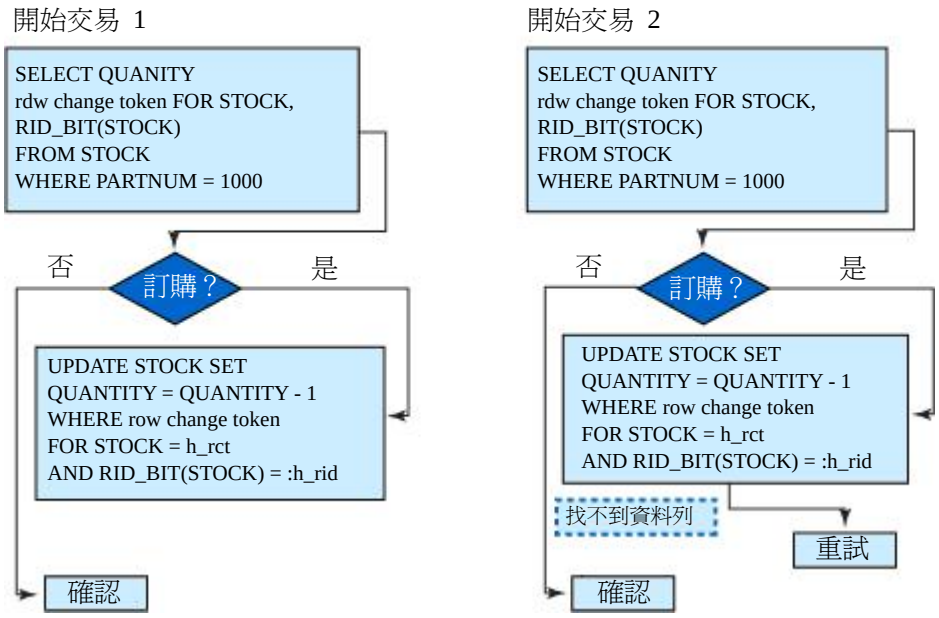
- **時間型更新偵測：**使用 ROW CHANGE TIMESTAMP 表示式將這項功能新增至 SQL。若要支援這項功能，就必須爲表格定義一個新產生的資料列變更時間戳記直欄，以儲存時間戳記值。這個直欄可以使用 ALTER TABLE 陳述式新增至現有表格，或在建立新表格時定義資料列變更時間戳記直欄。資料列變更時間戳記直欄的存在也會影響樂觀鎖定的行爲，因爲該直欄可用來將資料列變更記號的精細度從頁面層級提高到資料列層級，如此可以使樂觀鎖定應用程式大大獲益。

- **隱式隱藏直欄：**爲了解決相容性的問題，這項功能簡化了將資料列變更時間戳記直欄納入現有表格和應用程式的工作。使用隱式直欄清單時，隱式隱藏直欄不會外部化。例如，對表格執行 SELECT * 並不會在結果表格中傳回隱式隱藏直欄，而沒有直欄清單的 INSERT 陳述式預期不會得到隱式隱藏直欄的值，但應定義該直欄可容許空值或其他預設值。

使用此程式設計模型的應用程式將因加強的樂觀鎖定功能而獲益。請注意，未使用此程式設計模型的應用程式並不算是樂觀鎖定應用程式，而且會繼續依照舊有方式運作。

圖 3 說明 DB2 9.5 樂觀鎖定功能的運作方式。交易 1 和 交易 2 讀取同一個資料列，包括 RID_BIT 和 ROW CHANGE TOKEN 的值。然後交易 1 會在確定該資料列在前次執行 SELECT 並未變更之後，新增 RID_BIT 和 ROW CHANGE TOKEN 述詞至 UPDATE 陳述式來更新該資料列。現在，當交易 2 嘗試使用和交易 1 相同的述詞來更新同一個資料列時，將會找不到該資料列，因爲交易 1 的 UPDATE 關係，ROW CHANGE TOKEN 的值已變更。交易 2 必須啓動重試作業，才能擷取現行資料。

圖 3. DB2 9.5 加強的樂觀鎖定功能



啓用樂觀鎖定功能

因爲不用變更相關表格的 DDL 也能使用新的 SQL 表示式和樂觀鎖定功能的屬性，因此您可以在測試應用程式中輕鬆嘗試樂觀鎖定功能。

請注意，和變更 DDL 比起來，在沒有變更 DDL 的情況下，樂觀鎖定應用程式可能會發生更多負向誤判 (false negative) 的情形。真的會發生負向誤判的應用程式在正式作業環境中可能會無法順利延展，因爲負向誤判可能會造成重試的次數過多。因此，爲了避免發生負向誤判的情形，樂觀鎖定的目標表格應爲下列之一：

- 建立時便包含 ROW CHANGE TIMESTAMP 直欄
- 經過變更以包含 ROW CHANGE TIMESTAMP 直欄

爲了在您的應用程式中啓用樂觀鎖定支援，您必須執行一些基本步驟：

- 在起始查詢中，針對您必須處理的每個資料列執行 SELECT 資料列識別碼（使用 RID_BIT() 和 RID() 內建函數）以及資料列變更記號。
- 解除對資料列的鎖定，以便其他應用程式可以針對表格進行 SELECT、INSERT、UPDATE 及 DELETE 等作業（例如使用隔離層次游標穩定性或未確認讀取）。
- 在搜尋條件中使用資料列識別碼和資料列變更記號，在目標資料列上執

行搜尋的 UPDATE 或 DELETE 作業，樂觀地假定自從原始的 SELECT 陳述式以來，該解除鎖定的資料列並未變更。
- 如果該資料列已變更，UPDATE 作業將會失敗，而應用程式邏輯必須處理該失敗作業。例如，應用程式會重試 SELECT 和 UPDATE 作業。

執行上述步驟之後：

- 如果您的應用程式所執行的重試次數似乎高過了預期或所要的次數，那麼請新增資料列變更時間戳記直欄至您的表格，以確保只有經過 RID_BIT 函數識別的資料列變更僅會使資料列變更記號無效，而不會影響同一個資料頁面上的其他活動。
- 若要查看特定時間範圍內所插入或更新的資料列，請建立或變更表格，使其包含資料列變更時間戳記直欄。這個直欄將會由資料庫管理程式自動維護，而且可以使用直欄名稱或 ROW CHANGE TIMESTAMP 表示式進行查詢。
- 如果該直欄是使用 IMPLICITLY HIDDEN 屬性定義，那麼當表格直欄的參照爲隱式時，該直欄並不會外部化（僅適用於資料列變更時間戳記直欄）。但是，隱式隱藏直欄一律可以在 SQL 陳述式中明確地參照。這對於將直欄新增至表格會造成使用隱式直欄清單的現有應用程式失敗時非常有用。

資料列變更記號的精細度和負向誤判

RID_BIT() 內建函數和資料列變更記號是進行樂觀鎖定的唯一要求。但是，表格的綱目也會影響樂觀鎖定的行爲。

例如，資料列變更時間戳記直欄會使得 DB2 伺服器儲存前次變更（或初始插入）資料列的時間。如此提供了一個可以擷取最近一次變更資料列的時間戳記的方法。資料列變更時間戳記直欄是使用下列其中一種陳述式子句來定義：

- **GENERATED ALWAYS** FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP
  這個時間戳記直欄一律由資料庫管理程式維護

- **GENERATED BY DEFAULT** FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP
  這個時間戳記直欄依預設是由資料庫管理程式維護，但也接受使用者提供的輸入值。

當應用程式在表格上使用新的 ROW CHANGE TOKEN 表示式時，有兩種可能性必須考慮：

- 表格沒有資料列變更時間戳記直欄：ROW CHANGE TOKEN 表示式會傳回衍生的 BIGINT 值，這個值會由位於同一頁面上的所有資料列共用。如果頁面上的某個資料列經過更新，則同一頁面上所有資料列的資料列變更記號就會變更。這代表對其他資料列進行變更時，更新會失敗，這種特性稱爲負向誤判。

  附註：只有在應用程式可以忍受負向誤判，而且不會新增額外的儲存體至 ROW CHANGE TIMESTAMP 直欄的每個資料列時，才使用這個模式。

- 表格有資料列變更時間戳記直欄：ROW CHANGE TOKEN 表示式會傳回由直欄中時間戳記值所衍生的 BIGINT 值。在此情況下，負向誤判可能會發生，但是頻率會較低：如果表格經過重組或重新分送，在資料列移動且應用程式使用了先前的 RID_BIT() 值時，就可能會發生負向誤判。

資料列變更時間戳記直欄是否存在可以透過下列 SELECT 來查詢：

清單 1. SELECT 查詢

```
SELECT COLNAME, ROWCHANGETIMESTAMP, GENERATED FROM SYSCAT.COLUMNS
     WHERE TABNAME='tablename' AND ROWCHANGETIMESTAMP='Y'

COLNAME      ROWCHANGETIMESTAMP    GENERATED
-----------  ------------------    ---------
ROWCHGTS     Y                 A
```

在此範例中，存在一個資料列變更時間戳記直欄 ROWCHGTS，而且是以 GENERATED ALWAYS 子句所定義（"A" 值代表 GENERATED ALWAYS，"D" 值則代表 GENERATED BY DEFAULT）

時間型更新偵測

部分應用程式必須知道特定時間範圍的資料庫更新項目，這項資訊可以用於抄寫

資料、審核情境實例等。這會透過包含資料列變更時間戳記直欄的表格來實作，該直欄經過定義，可儲存由 ROW CHANGE TIMESTAMP 表示式所產生的時間戳記值。這個新的 ROW CHANGE TIMESTAMP 表示式會傳回一個代表前次變更資料列時間的時間戳記，並以類似 CURRENT TIMESTAMP 的當地時間表示。對於已經過更新的資料列而言，這會反映最近一次更新資料列的時間。否則，該值會與原始插入該資料列的時間相對應。

自 ALTER TABLE 陳述式之後尚未經過更新的資料列則會傳回直欄的類型預設值，該值爲 1 年 1 月 1 日午夜。只有經過更新的資料列才會有唯一的時間戳記。使用離線表格重組具體化時間戳記的資料列會傳回在表格重組期間所產生的唯一時間戳記。使用 INPLACE 選項的 REORG 並不足夠，因爲它不會使綱目變更具體化。

清單 2. 建立包含資料列變更時間戳記直欄的表格

```
CREATE TABLE EMPLOYEE (EMPNO CHAR(6) NOT NULL,
       ......
       ROWCHGTS TIMESTAMP NOT NULL
       GENERATED ALWAYS
       FOR EACH ROW ON UPDATE AS
       ROW CHANGE TIMESTAMP)
```

清單 3. 建立未包含資料列變更時間戳記直欄的表格，但稍後透過 ALTER TABLE 陳述式新增該直欄

```
ALTER TABLE EMPLOYEE ADD COLUMN
       ROWCHGTS TIMESTAMP NOT NULL
       GENERATED ALWAYS
       FOR EACH ROW ON UPDATE AS
       ROW CHANGE TIMESTAMP
```

清單 4. 選取最近 30 天內已變更的所有資料列

```
SELECT * FROM EMPLOYEE WHERE
       ROW CHANGE TIMESTAMP FOR EMPLOYEE <= CURRENT TIMESTAMP AND
```

```
        ROW CHANGE TIMESTAMP FOR EMPLOYEE >= CURRENT TIMESTAMP - 30 days
```

表 1 建立包含資料列變更時間戳記直欄的表格時，利用 INSERT、IMPORT 或 LOAD 填入資料後的 ROW CHANGE TIMESTAMP 直欄內容

EMPNO	FIRSTNME	LASTNAME	PHONENO	ROW CHANGE TIMESTAMP
000010	CHRISTINE	HAAS	3978	2007-12-20 13:53:01.296000
000020	MICHAEL	THOMPSON	3476	2007-12-20 13:53:01.312000
000030	SALLY	KWAN	4738	2007-12-20 13:53:01.312001

表 2 將資料列變更時間戳記直欄新增至現有表格時的 ROW CHANGE TIMESTAMP 直欄內容

EMPNO	FIRSTNME	LASTNAME	PHONENO	ROW CHANGE TIMESTAMP
000010	CHRISTINE	HAAS	3978	0001-01-01 00:00:00.000000
000020	MICHAEL	THOMPSON	3476	0001-01-01 00:00:00.000000
000030	SALLY	KWAN	4738	0001-01-01 00:00:00.000000

隱式隱藏直欄

這個功能簡化了將資料列變更時間戳記直欄納入現有表格及應用程式的工作。除非是以名稱明確參照該直欄，否則 CREATE 或 ALTER TABLE 陳述式上的 IMPLICITLY HIDDEN 屬性可指定在 SQL 陳述式中看不見該直欄。例如，假定表格包含使用 IMPLICITLY HIDDEN 子句定義的直欄，SELECT * 的結果並不會包含隱式隱藏直欄。但是，明確參照隱式隱藏直欄名稱的 SELECT 結果將會在結果表格中包含該直欄。您只能針對 ROW CHANGE TIMESTAMP 直欄指定 IMPLICITLY HIDDEN。

清單 5. 資料列變更時間戳記的隱式隱藏直欄

```
    CREATE TABLE SALARY_INFO (
           LEVEL INT NOT NULL,
           SALARY INT NOT NULL,
           UPDATE_TIME TIMESTAMP NOT NULL
           IMPLICITLY HIDDEN
           GENERATED ALWAYS FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP)

             or
```

```
ALTER TABLE SALARY_INFO
ADD COLUMN UPDATE_TIME TIMESTAMP NOT NULL
IMPLICITLY HIDDEN
GENERATED ALWAYS FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP
```

清單 6. 使用 DESCRIBE 指令顯示表格的直欄

```
DESCRIBE TABLE SALARY_INFO

           Data type                  Column
Column name  schema    Data type name     Length    Scale Nulls
-----------  --------- ------------------ ---------- ----- ------
LEVEL       SYSIBM    INTEGER                 4     0 No
SALARY      SYSIBM    INTEGER                 4     0 No
UPDATE_TIME  SYSIBM   TIMESTAMP              10    0 No
```

清單 7. 隱式隱藏直欄的 INSERT 和 SELECT

```
INSERT INTO SALARY_INFO VALUES (1, 50000)

SELECT * FROM SALARY_INFO

LEVEL      SALARY
---------- -----------
         1      50000
```

清單 8. 明確參照隱式隱藏直欄的 INSERT 和 SELECT

```
INSERT INTO SALARY_INFO (LEVEL, SALARY, UPDATE_TIME)
       VALUES (2, 30000, DEFAULT)

SELECT LEVEL, SALARY, UPDATE_TIME
       FROM SALARY_INFO
       WHERE LEVEL = 2
```

```
LEVEL      SALARY     UPDATE_TIME
---------- ---------- --------------------------
         2      30000 2007-12-18-15.34.24.437000
```

樂觀鎖定的限制和考量

- 主要索引鍵、外部索引鍵、多維度叢集 (MDC) 直欄、範圍分割區直欄、資料庫雜湊分割鍵、DETERMINED BY 限制直欄和暱稱均不支援 ROW CHANGE TIMESTAMP 直欄。
- 資料庫分割功能 (DPF) 配置不支援 RID() 函數。
- 在樂觀鎖定情境實例中的提取和更新作業之間執行的線上或離線表格 REORG，可能會造成更新失敗，但這應由一般的應用程式重試邏輯來處理。
- 在 9.5 版中，IMPLICITLY HIDDEN 屬性僅限用於樂觀鎖定 ROW CHANGE TIMESTAMP 直欄。
- 如果表格的 ROW CHANGE TIMESTAMP 直欄是以新增至現有表格的方式加入，那麼 INPLACE REORG 會受到限制，直到保證所有資料列都已具體化爲止（會針對這個錯誤傳回 SQL2219，原因碼 13）。這可以使用 LOAD REPLACE 指令或典型的表格 REORG 來完成，如此可以避免正向誤判。建立包含 ROW CHANGE TIMESTAMP 直欄的表格則沒有任何限制。

使用實務

某位員工被分配到新的職責，因此現在爲另一個部門工作。兩位經理（舊部門經理稱爲 Manager1，而新部門經理稱爲 Manager2）正在使用人員管理應用程式更新 SAMPLE 資料庫 EMPLOYEE 表格中的員工記錄，有可能兩位經理會同時嘗試更新同一位員工的記錄。當 Manager1 選取並更新該員工的記錄時，Manager2 也更新同一個記錄。舉例來說，「樂觀鎖定」功能可讓 Manager2 在目前的應用程式更新之前知道特定記錄已經過更新。如此一來，此應用程式的設計更容易，因爲它不需要實作其本身的更新偵測邏輯。

情境 1

EMPLOYEE 表格包含隱式隱藏的 ROW CHANGE TIMESTAMP 直欄（之前新增的），而且只有 Manager1 存取該表格。Manager1 從 EMPLOYEE 表格選取資料，並於稍後嘗試將 Christine Haas 的電話號碼從 3978 更新爲 1092。更新成功。

清單 9. SELECT 陳述式 (Manager1)

```
SELECT RID_BIT(EMPLOYEE),
       ROW CHANGE TOKEN FOR EMPLOYEE,
       EMPNO, FIRSTNME, LASTNAME, PHONENO, ROWCHGTS
       FROM EMPLOYEE FETCH FIRST 3 ROWS ONLY
```

表 3. SELECT 的結果

樂觀鎖定表示式		EMPLOYEE 表格				
RID_BIT	**ROW CHANGE TOKEN**	**EMPNO**	**FIRSTNME**	**LASTNAME**	**PHONENO**	**ROW CHANGE TIMESTAMP**
x'0400400100000000 0000000000FA9023'	74904229642240	000010	CHRISTINE	HAAS	3978	0001-01-01 00:00:00.000000
x'0500400100000000 0000000000FA9023'	74904229642240	000020	MICHAEL	THOMPSON	3476	0001-01-01 00:00:00.000000
x'0600400100000000 0000000000FA9023'	74904229642240	000030	SALLY	KWAN	4738	0001-01-01 00:00:00.000000

清單 10. UPDATE 陳述式

```
UPDATE EMPLOYEE SET
     (FIRSTNME,LASTNAME,PHONENO) = ('CHRISTINE','HAAS','1092')
     WHERE RID_BIT(EMPLOYEE)=x'040040010000000000000000000FA9023' AND
     ROW CHANGE TOKEN FOR EMPLOYEE=74904229642240
```

表 4. UPDATE 的結果

RID_BIT	ROW CHANGE TOKEN	EMPNO	FIRSTNME	LASTNAME	PHONENO	ROW CHANGE TIMESTAMP
x'0400400100000000 0000000000FA9023'	***141285645885181032***	000010	CHRISTINE	HAAS	1092	***2007-12-20 11:55:45.593000***
x'0500400100000000 0000000000FA9023'	74904229642240	000020	MICHAEL	THOMPSON	3476	0001-01-01 00:00:00.000000
x'0600400100000000	74904229642240	000030	SALLY	KWAN	4738	0001-01-01

0000000000FA9023'						00:00:00.000000

情境 2

EMPLOYEE 表格包含隱式隱藏的 ROW CHANGE TIMESTAMP 直欄，而且 Manager1 和 Manager2 同時存取該表格。Manager1 從 EMPLOYEE 表格選取資料，並於稍後嘗試更新同一筆資料。但是，從他選取到更新資料的這段時間，Manager2 更新了同一筆資料。Manager2 的更新成功，但 Manager1 的更新失敗。

表 5. SELECT 的結果（Manager1 和 Manager2）

RID_BIT	ROW CHANGE TOKEN	EMPNO	FIRSTNME	LASTNAME	PHONENO	ROW CHANGE TIMESTAMP
x'0400400100000000 0000000000FA9023'	74904229642240	000010	CHRISTINE	HAAS	3978	0001-01-01 00:00:00.000000
x'0500400100000000 0000000000FA9023'	74904229642240	000020	MICHAEL	THOMPSON	3476	0001-01-01 00:00:00.000000
x'0600400100000000 0000000000FA9023'	74904229642240	000030	SALLY			

清單 11. UPDATE 陳述式 (Manager2)

```
UPDATE EMPLOYEE SET
     (FIRSTNME,LASTNAME,PHONENO) = ('CHRISTINE','HAAS','1092')
     WHERE RID_BIT(EMPLOYEE)=x'040040010000000000000000FA9023' AND
     ROW CHANGE TOKEN FOR EMPLOYEE=74904229642240
```

表 6. UPDATE 的結果 (Manager2)

RID_BIT	ROW CHANGE TOKEN	EMPNO	FIRSTNME	LASTNAME	PHONENO	ROW CHANGE TIMESTAMP
x'0400400100000000 0000000000FA9023'	***141285645885181032***	000010	CHRISTINE	HAAS	1092	***2007-12-20 11:55:45.593000***
x'0500400100000000 0000000000FA9023'	74904229642240	000020	MICHAEL	THOMPSON	3476	0001-01-01 00:00:00.000000

x'0600400100000000 0000000000FA9023'	74904229642240	000030	SALLY	KWAN	4738	0001-01-01 00:00:00.000000

清單 12. UPDATE 陳述式 (Manager1)

```
UPDATE EMPLOYEE SET
     (FIRSTNME,LASTNAME,PHONENO) = ('CHRISTINE','HAAS','1092')
     WHERE RID_BIT(EMPLOYEE)=x'040040010000000000000000FA9023' AND
     ROW CHANGE TOKEN FOR EMPLOYEE=74904229642240
```

UPDATE 的結果 (Manager1)

Manager1 的更新沒有成功。因爲 Manager2 的 UPDATE 讓 ROW CHANGE TOKEN 產生變更，在比較 SELECT 時所擷取的記號和被 Manager2 的應用程式更新之後的現行值時，Manager1 UPDATE 陳述式的 ROW CHANGE TOKEN 述詞會失敗。因此，UPDATE 無法找到指定的資料列。您會收到 "SQL0100W No row was found for FETCH, UPDATE or DELETE; or the result of a query is an empty table. SQLSTATE=02000" 的訊息。

情境 3

EMPLOYEE 表格包含隱式隱藏的 ROW CHANGE TIMESTAMP 直欄，而且 Manager1 和 Manager2 同時存取該表格。Manager1 更新了一些資料列，但尚未確定其變更。Manager2 以「未確認讀取」(Uncommitted Read) 隔離層次從 Employee 表格選取了該資料。Manager1 確定其變更。Manager2 嘗試更新同一筆資料。Manager2 的最終更新成功，因爲其讀取的是 Manager1 未確認更新項目。但是，如果 Manager1 回復而不是確認該更新作業，Manager2 的更新就會失敗。

表 7. SELECT 的結果 (Manager1)

RID_BIT	ROW CHANGE TOKEN	EMPNO	FIRSTNME	LASTNAME	PHONENO	ROW CHANGE TIMESTAMP
x'0400400100000000 0000000000FA9023'	74904229642240	000010	CHRISTINE	HAAS	3978	0001-01-01 00:00:00.000000
x'0500400100000000	74904229642240	000020	MICHAEL	THOMPSON	3476	0001-01-01

0000000000FA9023'						00:00:00.000000
x'0600400100000000 0000000000FA9023'	74904229642240	000030	SALLY	KWAN	4738	0001-01-01 00:00:00.000000

清單 13. 未確認的 UPDATE 陳述式 (Manager1)

```
UPDATE EMPLOYEE SET
     (FIRSTNME,LASTNAME,PHONENO) = ('CHRISTINE','HAAS','1092')
     WHERE RID_BIT(EMPLOYEE)=x'040040010000000000000000000FA9023' AND
     ROW CHANGE TOKEN FOR EMPLOYEE=74904229642240
```

表 8. SELECT 的結果 (Manager2)

RID_BIT	ROW CHANGE TOKEN	EMPNO	FIRSTNME	LASTNAME	PHONENO	ROW CHANGE TIMESTAMP
x'0400400100000000 0000000000FA9023'	***141285665533242120***	000010	CHRISTINE	HAAS	1092	***2007-12-20 16:47:03.125000***
x'0500400100000000 0000000000FA9023'	74904229642240	000020	MICHAEL	THOMPSON	3476	0001-01-01 00:00:00.000000
x'0600400100000000 0000000000FA9023'	74904229642240	000030	SALLY	KWAN	4738	0001-01-01 00:00:00.000000

表 9. 確認 UPDATE 的結果 (Manager1)

RID_BIT	ROW CHANGE TOKEN	EMPNO	FIRSTNME	LASTNAME	PHONENO	ROW CHANGE TIMESTAMP
x'0400400100000000 0000000000FA9023'	***141285665533242120***	000010	CHRISTINE	HAAS	1092	***2007-12-20 16:47:03.125000***
x'0500400100000000 0000000000FA9023'	74904229642240	000020	MICHAEL	THOMPSON	3476	0001-01-01 00:00:00.000000
x'0600400100000000 0000000000FA9023'	74904229642240	000030	SALLY	KWAN	4738	0001-01-01 00:00:00.000000

清單 14. UPDATE 陳述式 (Manager2)

```
UPDATE EMPLOYEE SET
```

```
        (FIRSTNME,LASTNAME,PHONENO) = ('CHRISTINE','HAAS','1090')
        WHERE RID_BIT(EMPLOYEE)=x'040040010000000000000000000FA9023' AND
        ROW CHANGE TOKEN FOR EMPLOYEE=141285665533242120
```

表 10. UPDATE 的結果 (Manager2)

RID_BIT	ROW CHANGE TOKEN	EMPNO	FIRSTNME	LASTNAME	PHONENO	ROW CHANGE TIMESTAMP
x'0400400100000000 0000000000FA9023'	***141285667099502664***	000010	CHRISTINE	HAAS	1090	***2007-12-20 16:51:53.125000***
x'0500400100000000 0000000000FA9023'	74904229642240	000020	MICHAEL	THOMPSON	3476	0001-01-01 00:00:00.000000
x'060040010000000000 00000000FA9023'	74904229642240	000030	SALLY	KWAN	4738	0001-01-01 00:00:00.000000

Manager1 確認其變更而 Manager2 嘗試更新同一筆資料的結果：

因爲 Manager2 讀取的是 Manager1 未確認的更新，所以 Manager2 的最終更新成功；且因爲 Manager1 確認該變更會產生新的記號，因此 Manager2 UPDATE 陳述式中的 ROW CHANGE TOKEN 述詞就會成功。

表 11. Manager1 發出 ROLLBACK 而不是 COMMIT 的結果

RID_BIT	ROW CHANGE TOKEN	EMPNO	FIRSTNME	LASTNAME	PHONENO	ROW CHANGE TIMESTAMP
x'0400400100000000 0000000000FA9023'	***74904229642240***	000010	CHRISTINE	HAAS	3978	***0001-01-01 00:00:00.000000***
x'0500400100000000 0000000000FA9023'	74904229642240	000020	MICHAEL	THOMPSON	3476	0001-01-01 00:00:00.000000
x'0600400100000000 0000000000FA9023'	74904229642240	000030	SALLY	KWAN	4738	0001-01-01 00:00:00.000000

在 Manager1 回復其變更之後，截至 Manager1 未確認 UPDATE 前嘗試更新資料之 Manager2 的結果：

Manager2 的最終更新沒有成功，因爲 Manager1 已回復至原始記號，因此 UPDATE 無法找到該資料列，所以導致 ROW CHANGE TOKEN 述詞失敗。

情境 4

EMPLOYEE 表格沒有 ROW CHANGE TIMESTAMP 直欄，而且 Manager1 和 Manager2 同時存取該表格。Manager1 選取了一個資料列並嘗試加以更新。但是，從他選取到更新這個資料列的這段時間，Manager2 更新了同一個資料頁面上的其他資料（並不一定是要和 Manager1 一樣的資料，但是在另一個資料列裡面）。因此，當 Manager1 嘗試更新資料時，更新會失敗。

清單 15. UPDATE 陳述式 (Manager2)

```
UPDATE EMPLOYEE SET
     (FIRSTNME,LASTNAME,PHONENO) = ('CHRISTINE','HAAS','1092')
     WHERE RID_BIT(EMPLOYEE)=x'040040010000000000000000FA9023' AND
     ROW CHANGE TOKEN FOR EMPLOYEE=74904229642240
```

表 12. UPDATE 的結果 (Manager2)

RID_BIT	ROW CHANGE TOKEN	EMPNO	FIRSTNME	LASTNAME	PHONENO
x'0400400100000000 0000000000FA9023'	***141285645885181032***	000010	CHRISTINE	HAAS	1092
x'0500400100000000 0000000000FA9023'	***141285645885181032***	000020	MICHAEL	THOMPSON	3476
x'0600400100000000 0000000000FA9023'	***141285645885181032***	000030	SALLY	KWAN	4738

清單 16. 另一個資料列的 UPDATE 陳述式 (Manager1)

```
UPDATE EMPLOYEE SET
     (FIRSTNME,LASTNAME,PHONENO) = ('MICHAEL','THOMPSON','9012')
     WHERE RID_BIT(EMPLOYEE)=x'040040010000000000000000FA9023' AND
     ROW CHANGE TOKEN FOR EMPLOYEE=74904229642240
```

Manager1 嘗試更新同一個資料頁面上的另一個資料列的結果：

Manager1 的更新沒有成功，因爲即使 Manager1 嘗試更新的資料列實際上並未變更，但是所有資料列的 ROW CHANGE TOKEN 值卻已變更，所以在比較記

號時，ROW CHANGE TOKEN 述詞會失敗。如果之前在 EMPLOYEE 表格中加入資料列變更時間戳記直欄，這個負向誤判情境便不會使 UPDATE 造成失敗。

情境 5

EMPLOYEE 表格沒有 ROW CHANGE TIMESTAMP 直欄。該表格經過變更並加入了 ROW CHANGE TIMESTAMP 直欄。Manager1 和 Manager2 存取同一個表格。Manager1 選取了一個資料列並嘗試加以更新。但是，從他選取到更新這個資料列的這段時間，Manager2 更新了同一個資料頁面上的其他資料（並不一定是要和 Manager1 一樣的資料，但是在另一個資料列裡面）。由於已新增了 ROW CHANGE TIMESTAMP 直欄，因此即使這些資料列位於同一個頁面上，更新不同的資料列還是都會成功。

清單 17. UPDATE 陳述式 (Manager2)

```
UPDATE EMPLOYEE SET
     (FIRSTNME,LASTNAME,PHONENO) = ('CHRISTINE','HAAS','1092')
     WHERE RID_BIT(EMPLOYEE)=x'04004001000000000000000000FA9023' AND
     ROW CHANGE TOKEN FOR EMPLOYEE=74904229642240
```

表 13. UPDATE 的結果 (Manager2)

RID_BIT	ROW CHANGE TOKEN	EMPNO	FIRSTNME	LASTNAME	PHONENO	ROW CHANGE TIMESTAMP
x'0400400100000000 0000000000FA9023'	***141285673714388072***	000010	CHRISTINE	HAAS	1092	***2007-12-20 18:22:25.593000***
x'0500400100000000 0000000000FA9023'	74904229642240	000020	MICHAEL	THOMPSON	3476	0001-01-01 00:00:00.000000
x'0600400100000000 0000000000FA9023'	74904229642240	000030	SALLY	KWAN	4738	0001-01-01 00:00:00.000000

清單 18. 另一個資料列的 UPDATE 陳述式 (Manager1)

```
UPDATE EMPLOYEE SET
     (FIRSTNME,LASTNAME,PHONENO) = ('MICHAEL','THOMPSON','9012')
     WHERE RID_BIT(EMPLOYEE)=x'04004001000000000000000000FA9023' AND
```

```
          ROW CHANGE TOKEN FOR EMPLOYEE=74904229642240
```

表 14. UPDATE 的結果 (Manager1)

RID_BIT	ROW CHANGE TOKEN	EMPNO	FIRSTNME	LASTNAME	PHONENO	ROW CHANGE TIMESTAMP
x'0400400100000000 0000000000FA9023'	***141285673714388072***	000010	CHRISTINE	HAAS	1092	***2007-12-20 18:22:25.593000***
x'0500400100000000 0000000000FA9023'	***141285673726689984***	000020	MICHAEL	THOMPSON	9012	***2007-12-20 18:22:37.312000***
x'0600400100000000 0000000000FA9023'	74904229642240	000030	SALLY	KWAN	4738	0001-01-01 00:00:00.000000

情境 6

EMPLOYEE 表格有 ROW CHANGE TIMESTAMP 直欄，而且只有 Manager1 存取該表格。Manager1 選取了一些資料列並嘗試加以更新。但是，從他選取到更新這些資料列的這段時間，該表格已離線重組過。之後，當 Manager1 嘗試更新資料時，更新會失敗。這些資料列不會被更新，因爲 REORG 的結果，所以 ROW CHANGE TIMESTAMP 直欄已變更。

清單 19. 重組 EMPLOYEE 表格

```
    REORG TABLE EMPLOYEE
```

表 15. 重組 EMPLOYEE 表格之後的變更

RID_BIT	ROW CHANGE TOKEN	EMPNO	FIRSTNME	LASTNAME	PHONENO	ROW CHANGE TIMESTAMP
x'0400400100000000 0000000000FA9023'	***141285781563232400***	000010	CHRISTINE	HAAS	3978	***2007-12-21 11:29:30.250000***
x'0500400100000000 0000000000FA9023'	***141285781563232401***	000020	MICHAEL	THOMPSON	3476	***2007-12-21 11:29:30.250001***
x'0600400100000000 0000000000FA9023'	***141285781563232402***	000030	SALLY	KWAN	4738	***2007-12-21 11:29:30.250002***

清單 20. UPDATE 陳述式，因爲已執行表格 REORG，因此會在之後執行本陳述式 (Manager1)

```
UPDATE EMPLOYEE SET
     (FIRSTNME,LASTNAME,PHONENO) = ('CHRISTINE','HAAS','1092')
     WHERE RID_BIT(EMPLOYEE)=x'040040010000000000000000000FA9023' AND
     ROW CHANGE TOKEN FOR EMPLOYEE=74904229642240
```

在執行 REORG 之後，Manager1 嘗試更新資料的結果：

Manager1 的更新沒有成功，因爲從 Manager1 SELECT 到 UPDATE 的這段時間，另一個作業已離線重組過該表格，所以比較在 SELECT 時所擷取的記號和目前的記號，ROW CHANGE TOKEN 述詞會失敗。因此，UPDATE 陳述式無法使用在 REORG 發生之前所擷取的 ROW CHANGE TOKEN 來找到該資料列。

總結

爲了避免使用悲觀鎖定策略時，可能發生資料遭到鎖定而必須等待的情況，DB2 9.5 的樂觀鎖定功能將其他交易無法使用特定資源的時間縮到最短。因爲資料庫管理程式可以確定變更資料列的時間，因此可以確保資料的完整性，同時限制保持鎖定的時間。使用樂觀並行控制，資料庫管理程式便可以在讀取作業完成之後，立即解除資料列或頁面的鎖定。

DB2 9.5 新增的支援可輕鬆快速地進行樂觀鎖定，而不會發生正向誤判的情形。這項支援是使用資料列識別碼（RID_BIT 或 RID）內建函數、ROW CHANGE TOKEN 表示式、時間型更新偵測及隱式隱藏直欄來新增的。使用此程式設計模型的應用程式會因加強的樂觀鎖定功能而獲益並提高並行性。

關於作者

Werner Schuetz 是 IBM 認證的 IT 專家、IBM 認證的高階資料庫管理員，以及認證的 DB2 9 for Linux, UNIX, and Windows 應用程式開

發人員。他在德國 Stuttgart 的 IBM 創新中心擔任 DB2 技術顧問一職。IBM 創新中心提供獨立軟體供應商 (ISV) 跨平台的技術應用啓用支援，以進行植入、測試及移轉。在此背景下，Werner Schuetz 負責協助 ISV 測試 DB2 解決方案、執行效能與調整階段作業，以及執行競爭資料庫移轉。