



DB2いろはがるた



第6回 「へ」 - 並行性, 分離レベルで決定済み



執筆者

春野 さくら

「DB2いろはがるた」を執筆するために参上した、なぞの女性。日本の古典文学を愛する。

少し前に注目を集めたRed BrickというDBMS製品がありました。この製品は読み取り専用のデータベースだったので、ロッキングという仕組みがありませんでした。ロッキング・メカニズムがないので、データの挿入や更新ができないのですが、その代わりにパフォーマンスはすごくよかったです。しかし、多数の人が一つのデータベースを共通に使う普通の環境では、同時に同じデータを読み込んだり、更新したりすることがよく起こります。その場合、一番の問題は更新喪失と呼ばれている問題です。これは図のように、田中さんが飛行機の座席を予約しようとして座席予約表をチェックして7Cが空席であったので、予約したつもりが、山田さんがその後でその同じ席を予約してしまったという問題です。この結果田中さんは空港に到着して初めて、自分が予約したつもりの席に山田さんが座っているのを見て、怒り呆然となってしまいました。せっかくの楽しいバケーションの予定が、悲惨な休みになってしまいました。



そこで、普通データベース製品ではそういうことがおこらないように、更新を実行したトランザクションがコミットまたはロールバックしていない状態では、更新した行に対してXロックをかけます(コミットまたはロールバックされるまで、それは続きます)そして、Xロックのかかっている行に対しては、その他の人はXロックをとれないようにしています。つまり、更新することができないのです。このメカニズムによって前にお話したような悲劇をさけることができます。

次に、データの読み取りが先に行われる場合を考えてみましょう。田中さんはダラスに住んでいて、ホノルルにバケーションにいこうと計画しています。いいですね。アロハ！そこで、飛行機の座席予約状況をチェックしてみ

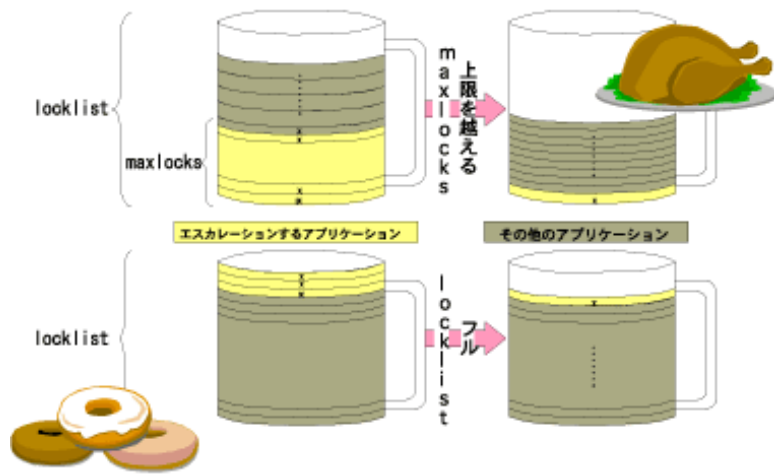
ると、ちょうどダラスからデンバー、デンバーからサンノゼ、サンノゼからホノルルへの便の座席があいていることがわかりました。(低料金のチケットなので大変です)そこで、最初にダラスからデンバーへの座席を予約することにしました。これは何なく予約できました。次にデンバーからサンノゼへの座席もOKでした。最後にサンノゼからホノルルへの便の予約をしようとして、あれれ、なんと先ほどあいていた座席が、ちょっとした間に、他の人に予約されていました。田中さんは、結局ホノルルの休暇がサンノゼの休暇になってしまいました。えらい違いですね。トホホ。(でもサンノゼもいいところです。歌にもあります。Do you know the way to San Jose?)。

FLIGHT	SEAT	NAME	DESTINATION	ORIGIN
512	7B	—	DENVER	DALLAS
⋮				
⋮				
814	8A	—	SAN JOSE	DENVER
⋮				
134	1C	—	HONOLULU	SAN JOSE
⋮				



どうしてこんなことになったのでしょうか？つまり最初にデータを読んだときに、そのデータを自分が更新するまで、他の人に更新されないための読み取りロックをかけていなかったせいです。DB2 UDBでは、こういった業務の目的に応じて、ロックの範囲を決める分離レベルというものをプログラムや文に対して設定することができます。前の例のように、最初に読み取った結果と、2度目に読んだときと同じ結果であることが必須である場合には、RRという分離レベルを指定します。そうすると、最初に呼んだ行に対してすべて読み取りロックであるSロックがとられ、他の人はその行に対してXロックをとることができません。つまり、他の人はSロックが解除されるまで、その行を更新することができないのです。最初に読んだ結果と比べて、2回目では新しい行が追加されてもよい場合には、RSという分離レベルを設定します。これを幻像読み取り(ファントム・リード)と呼びます。まあ、幻のような行が現れるからでしょう。最初と2度目の結果が厳密に同じでなくても、ある行を読んでいる時点の時だけ、他の人にその行を更新されたくない場合には、CS(カーソル固定)を設定します。この設定では最初と2回目の読み取り結果がちがうことが起こりえます。常に最新の状態の結果を取り出すからです。他の人がデータを更新してまだ最終的にコミットしていないような場合、もしかすると気が変わって変更するのをやめるかもしれないようなデータでも、そのデータを早く読み取りたい場合にはUR(ダーティ・リードとも呼ぶ)を設定します。(ダーティなんて、なんか失礼な呼びかたです)どの分離レベルを選ぶかはユーザーがどういうアプリケーションにしたいかという希望しだいなのです。それに対して、Oracleでは基本的に読み取りロックをとりません。その代わりに、読み取り一貫性というメカニズムをとっています。つまり、行の読み取りを始めた時点でのデータ一貫性を保証しています。読み取り中に他の人がそのデータを更新しているかもしれませんが、少なくともそのユーザーにとってはデータの整合性は保たれているわけです。

DB2 UDBにはロック・エスカレーションという機能が提供されています。行にロックをとると、36か72バイトの情報が1行について、メモリー上のlocklistと呼ばれる領域にとられます。そこで、全ロック情報がこの値を超えた時、または一つのアプリケーションがこのlocklistの中の%(maxlock)を越えた時、DB2は自動的に多数の行ロックを一つの表ロックに変更して、このロック情報のサイズを削減します。当然表ロックがかかりますので、その表に対する並行性はおちますが、現実にはメモリーは無限にあるわけではないのですから、これは非常に便利な機能といえるでしょう。locklist ,maxlockはDB構成パラメータで設定します。



その他にもlocktimeout DB構成パラメータで、指定した秒がたつと自動的にロックウェイト状態を解消するように設定することができます。デッドロックを自動的に検知する間隔も同じように指定することができます。こうやってみると、DB2 UDBはロックに関しては、非常に先進的で、かつ厳密、緻密に管理しているまじめすぎるぐらいのDBMS製品だといえますね。

[↑ 上に戻る](#)