データベースをより速くする

# 検索を早くする方法

データベースのテーブルにはたくさんの情報が格納されています。普段、私たちがSELECT文などでデータを取得するときは、この莫大な量を全て確認して必要な物のみを取得しています。そのためデータが増えれば増えるほどデータの取得に時間がかかってしまいます。

データベースにはこのように、処理が遅くならないようにインデックスと呼ばれる仕組みがあります。イメージは書籍などで、目的の内容がどのページに記載されているのか載っている索引(インデックス)と同様です。

# インデックスの作成と削除

データベースで作成されるインデックスには次のような特徴があります。

・インデックスは、指定した列に対して作られる。

・インデックスが存在する列に対して検索が行われた場合、自動的にインデックスの使用を

試みるため、高速になることが多い。

・場合によりインデックスが使用されずに速度向上につながらない時もある

・インデックスには名前を付けなければならない。

インデックスの基本構文

作成：CREATE INDEX インデックス名 ON 表名( 列名 ) ;

ALTER TABLE 表名 ADD INDEX インデックス名( 列名 ) ;

削除：DROP INDEX インデックス名 ON 表名 ;

ALTER TABLE 表名 DROP INDEX インデックス名 ;

ハンズオン　インデックスの作成と削除

１．従業員表のインデックス情報を確認する。

SHOW INDEX FROM EMPLOYEE ;

※この結果については、高速化のパターンで改めて説明します。

２．従業員表の入社日にインデックスを定義する。

CREATE INDEX EMP\_HID\_IDX ON EMPLOYEE( HIREDATE ) ;

SHOW INDEX FROM EMPLOYEE ;

３．入社日のインデックスを削除する。

DROP INDEX EMP\_HID\_IDX ON EMPLOYEE ;

SHOW INDEX FROM EMPLOYEE;

# 高速化のパターン

インデックスを作成すると、その列に関する検索が高速化します。では具体的にどのようなSQL文を実行すれば高速化するのか、３つのケースに分けて学習しましょう。

１．WHEREによる絞り込み

インデックスで指定した列を検索条件に使用すると、インデックスが使用されて

データ取得の高速化が望めます。ただし、部分一致検索や後方一致検索などでは

インデックスは使用されませんので注意が必要です。

２．ORDER BYによる並び替え

インデックスには並び替えを高速に行えるようにする効果があるためORDER BY

の処理も早くなります。

３．JOINによる結合の条件

結合処理は内部で並び替えを行っているため、インデックスのある列を使うと

高速になります。

このようにインデックスが作成されるとデータの検索が高速化します。これは以前学習した主キー制約やユニーク制約ですでに対象のデータが存在済みかチェックする場合にも利用できます。その為、主キー制約やユニーク制約、外部キー制約を作成すると自動的にインデックスも合わせて作成されます。先ほどのハンズオンですでにインデックスが存在していたのはこのためです。

また、外部キー制約を付ける場合には、参照先の列にインデックスの定義が必要です。

※ただし、学習環境のデータ件数程度では目に見えるほどの変化は感じられない。

# インデックスの注意点

インデックスを作成すると検索処理が高速化されますが、良い点だけではありません。

次にデメリットについて学習します。最大のデメリットは、インデックスを作成している表が変更されるとインデックス情報を更新する必要が出てくることです。もちろんインデックス情報は人間では無くデータベースが自動で更新してくれますが、頻繁に更新される場合データベースの負荷も高くなります。

データベースをより便利にする

# ビュー

データベースから情報を取得する時はSELECT文で欲しいデータの条件を指定してSQLを発行します。ただこの方法だとデータの取得を行うたびにSELECT文を作成する必要があります。単純なSQLであれば問題ないですが結合や副問い合わせなどがあると入力ミスなども発生しかねません。データベースには、結果表を一つの表のように扱える**ビュー(view)**という機能が備わっています。またビューは仮想表とも呼ばれます。

・ビューの基本構文

作成：CREATE VIEW ビュー名 AS SELECT文 ;

削除：DROP VIEW ビュー名 ;

・ビューのメリット

複雑なSQL文を簡潔に記述することが出来る。

セキュリティの高い情報(列)を隠蔽できる。

・従業員表 ・ビュー

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| EMP\_NO | ENAME | TEL | SALARY |  | EMP\_NO | ENMAE |
| 00001  00002  00003  00004 | 小林友治  小林圭輔  友保隼平  稲田直樹 | 0660030001  0660010001  0660010002  0660010003 | 5000  2875  950  900 |  | 00001  00002  00003  00004 | 小林友治  小林圭輔  友保隼平  稲田直樹 |

セキュリティの高い情報をビューに

することで隠蔽することが可能

ハンズオン　ビュー(仮想表)を作成する **※必須**

１．商品表と試作品表をまとめて表示するPRODVIEWビューを作成する。

CREATE VIEW PRODVIEW AS

SELECT PRODUCT\_NO, PNAME, CATEGORY, PRICE

FROM PRODUCT

UNION

SELECT PROTOTYPE\_NO, PROTONAME, CATEGORY, PRICE

FROM PROTOTYPE ;

２．PRODVIEWビューの情報を表示する。

SELECT \* FROM PRODVIEW ;

SHOW CREATE VIEW PRODVIEW ;

# ビューの制約とデメリット(P347)

ビューはテーブルと似ていますが、テーブルと同じわけではありません。例えば、テーブルは自由にINSERTやUPDATEを行うことが可能ですがビューに対してはいくつかの条件が揃わなければSELECTしか行うことが出来ません。これは、ビューがあくまでも**仮想的なテーブル**にすぎずデータを内部に持っていません。つまり、**ビューは単なる「名前を付けたSELECT文」**でしかありません。なので、**ビューはSQLを簡潔に書けても処理速度は速くならない**ので注意が必要です。

# 採番の方法(P349)

あるテーブルに行を追加する時に、主キーになにを設定するべきか迷うことがあります。主キーである以上、すでに使われている値と重複することは許されないので連番を振る方法がよく用いられます。追加する行に独自の番号を振るために、適切な番号を取得することを採番と言います。

採番の方式は、DBMSにより機能が分かれますので代表的な２種類を紹介します。

１．連番が自動的に振られる特殊な列を定義できる。(MySQLなど)

２．連番を管理してくれる専用の道具が提供されている。(Oracleなど)

ハンズオン　列に自動採番を定義する

１．自動採番を行う列を持ったテスト表を作成する。

CREATE TABLE TEST(

ID INT PRIMARY KEY **AUTO\_INCREMENT** ,

NAME VARCHAR(20)

) ;

２．データの登録を２件行う。

INSERT INTO TEST( NAME ) VALUES ( 'ECC 太朗' ) ;

INSERT INTO TEST( NAME ) VALUES ( 'ECC 花子' ) ;

３．テスト表のデータを確認する。

SELECT \* FROM TEST ;

４．テスト表の削除を行う。

DROP TABLE TEST ;