MySQL-進階物件

李偉銘

大綱

- View
  - 介紹
  - 新建View
  - 移除View
  - 修改View
  - 可異動資料的View

- Index
  - 介紹
  - 新建Index
  - 移除Index
  - 適用時機

View-介紹 (1/3)

- 簡述
  - 中譯檢視表，業界常以View表稱呼
  - 由一段select敘述描述而成，呈現的結果即select敘述的查詢結果
  - 邏輯上才存在的虛擬資料表(Virtual Table)，所以不太佔用硬碟空間
  - 單位層級第4層，與Table同層
  - 用在DQL時: 使用細節同Table
  - 用在DML時: 會受到限制，符合某些條件下才可使用，在後面章節說明
  - 因每次用到都會重新執行select敘述，所以須考慮到效能問題

View-介紹 (2/3)

- 基底資料表
  - 在定義View時，使用到的資料來源即為基底資料表(Based Table)
  - 一般狀況下，基底資料表即為Table，但其實也可以是另一個View
  - View是虛擬資料表，不會儲存資料，所以稱它是虛擬結構(Virtual Structure)
  - Table才是真正儲存資料的物件，所以Table是實體結構(Physical Structure)
  - 修改/移除基底資料表，可能會造成View產生錯誤!

View-介紹 (3/3)

- 常見應用
  - 簡化複雜的select敘述，程式端(EX. Java)對View操作即可
  - 當查詢資料需求變更時，更改View即可，而不用修改程式端
  - 隱藏資料，限制程式端工程師能看到的資料
  - 將View設計成唯讀，可避免程式端工程師修改資料

View-新建View (1/6)

- create view (1/2)
  - 語法

```
create [or replace] view 檢視表名 [(欄位名1, .., 欄位名N)]
as
select敘述..
[with [cascaded | local] check option]
```

  - 說明
    - 檢視表名: 新建的View名稱。業界習慣會在View名稱前加上V_，以跟Table區分
    - 欄位名: 呈現的欄位名稱。依序指定，須與select敘述的欄位數量符合
    - select敘述: 描述此View的select敘述

View-新建View (2/6)

- create view (2/2)
  - 範例

```
-- 範例-create view敘述-簡單View
-- 新建一個View，其內只包含部門編號30的員工資料
create view V_EMP30
as
select * from EMP where DEPTNO = 30;
```

View-新建View (3/6)

- 使用欄位名
  - 用來指定View呈現時的欄位名稱
  - 依序對應select敘述，且必須與select敘述的欄位數量相同

```
-- 範例-create view敘述-使用欄位名稱
-- 新建一個View，並指定呈現時的欄位名稱
create view V_EMP30(NO, NAME, TITLE, MGR_NO)
as
select EMPNO, ENAME, JOB, MGR from EMP where DEPTNO = 30;
```

NO	NAME	TITLE	MGR_NO
7499	ALLEN	SALESMAN	7698
7521	WARD	SALESMAN	7698
7654	MARTIN	SALESMAN	7698

View-新建View (4/6)

- 加上with [cascaded | local] check option (1/2)
  - 執行DML時，會先檢查是否符合描述View的where子句
  - 由於基底資料表也可能是View，所以可另外加上..
    - cascaded: (預設) 檢查全部
    - local: 只檢查當前View
  - 用在各DML敘述的狀況..
    - delete: 無差別
    - insert/update: 須保證執行完後，異動的資料還會出現在查詢結果，否則會出現錯誤訊息 CHECK OPTION failed '資料庫名.View名'
  - 若描述View的select敘述無where子句，加上with check option無作

View-新建View (5/6)

- 加上with [cascaded | local] check option (2/2)
  - 範例

```
-- 範例-create view敘述-加上with check option
-- 新建一個View，並加上with check option
create view V_EMP30
as
select * from EMP where DEPTNO = 30
with check option;
```

View-新建View (6/6)

```
-- 對有加上with check option的View執行insert敘述
insert into V_EMP30(EMPNO, ENAME, DEPTNO)
values (9999, 'William', 10);
-- 由於欲新增的資料部門編號為10，不會出現在V_EMP30查詢結果中
-- 所以新增會失敗，並看到錯誤訊息CHECK OPTION failed 'example.v_emp30'
```

```
-- 對有加上with check option的View執行update敘述
update V_EMP30 set DEPTNO = 20 where EMPNO = 7499;
-- 由於欲修改部門編號為20，就不會出現在V_EMP30查詢結果中
-- 所以修改會失敗，並看到錯誤訊息CHECK OPTION failed 'example.v_emp30'
```

View-移除View

- 語法

```
drop view [if exists] 檢視表名1 [, 檢視表名2, ..]
```

- 說明
  - 檢視表名: 欲移除的View名稱

*註: 由於View也有可能是其他View的基底資料表，
所以移除後可能造成其他View的錯誤!

- 範例

```
-- 範例-drop view敘述
-- 移除名為V_EMP30的View
drop view if exists V_EMP30;
```

View-修改View (1/2)

- 語法

```
alter view 檢視表名 [(欄位名1, .., 欄位名N)]
as
select敘述..
[with [cascaded | local] check option]
```

- 說明

*註

1. View本身是個不複雜的物件，若要修改View名稱，移除後重新建立即可
2. 由於View也有可能是其他View的基底資料表，所以修改後可能造成其他View的錯誤!

  - 檢視表名: 欲修改的View名稱
  - 欄位名: 呈現的欄位名稱。依序指定，須與select敘述的欄位數量符合
  - select敘述: 描述此View的select敘述
  - with [cascaded | local] check option: 執行DML時，檢查View本身where子句

View-修改View (2/2)

- 範例

```
-- 範例-alter view敘述
-- 修改名為V_EMP30的View
alter view V_EMP30
as
select * from EMP where DEPTNO = 30
    and MGR is not null;
```

View-可異動資料的View (1/3)

- 說明
  - 在 View-介紹 有提到，View可以用在DML，但會受到限制
  - 須符合某些條件下才可使用
  - 對View執行DML，會異動對應資料表內的資料
- 可安全異動資料的View
  - 須符合以下條件
  1. 無計算欄位、函數欄位
  2. 基底資料表只有一個

*註: 何謂計算欄位?
靠計算得到值的欄位
EX. 欄位 "年薪"，是用 "月薪*12" 算出，
而並非現成資料

View-可異動資料的View (2/3)

- View用在DML各敘述的限制
  - 可安全異動資料的View(前頁所述)，意義在於不會造成奇怪的資料
  - View用在insert、delete、update等敘述其實各有不同的限制
  - 這些限制較鬆，但可能造成奇怪的資料
  - EX. 基底資料表有2個(join)，其實仍可以執行update敘述，但修改結果可能跟預期不同
  - 可參考官方說明
    - https://dev.mysql.com/doc/refman/8.0/en/view-updatability.html

View-可異動資料的View (3/3)

- 範例

*註: 修改後，有3個員工的DNAME都變成"會計部"

```
-- 範例-可執行update敘述，但造成奇怪資料的View
-- 新建一個View，用到2個基底資料表
create view V_UPDATABLE as
select e.EMPNO, e.ENAME, d.DEPTNO, d.DNAME
from EMP e join DEPT d on e.DEPTNO = d.DEPTNO;

-- 執行update敘述
-- 修改員工編號7782的員工之部門名稱(DNAME)
update V_UPDATABLE
set DNAME = '會計部'
where EMPNO = 7782;
```

EMPNO	ENAME	DEPTNO	DNAME
7782	CLARK	10	會計部
7839	KING	10	會計部
7934	MILLER	10	會計部
7369	SMITH	20	RESEARCH
7566	JONES	20	RESEARCH
7788	SCOTT	20	RESEARCH
7876	ADAMS	20	RESEARCH
7902	FORD	20	RESEARCH
7499	ALLEN	30	SALES
7521	WARD	30	SALES
7654	MARTIN	30	SALES
7698	BLAKE	30	SALES
7844	TURNER	30	SALES
7900	JAMES	30	SALES

View-Exercise11

01.請新建一個View，名為V_DEPT_ECOUNT，列出部門編號、部門人數

02.請新建一個View，名為V_DEPT_ECOUNT2，基底資料表為第01題建立的檢視表V_DEPT_ECOUNT，並將欄位依序命名為DEPARTMENT_NO、EMPLOYEE_COUNT

03.請撰寫一select敘述，用第02題建立的檢視表V_DEPT_ECOUNT2，join資料表DEPT，列出部門名稱、部門人數

04.請新建一個View，名為V_EMP10，其內含有部門編號10的所有員工資料，並加上with check option限制DML

05.請修改檢視表V_DEPT_ECOUNT2，將欄位命名依序改為DEPT_NO、EMP_COUNT，其餘部分不變

06.請移除檢視表V_DEPT_ECOUNT2

Index-介紹 (1/3)

- 簡述
  - 中譯索引
  - 針對某個Table，由一個或多個欄位組成，且具有順序性
  - 會以平衡樹(B-Tree)結構儲存至一個檔案，佔用大量硬碟空間
  - 單位層級第5層，Table的下一層，與Column同層
  - Index就像書籍的目錄，通過目錄可以快速查詢到我們要看的那一頁
  - 執行DQL時: 透過Index查詢，效能會明顯提升
  - 執行DML時: 資料異動時，由於底層要重建Index，效能會下降

Index-介紹 (2/3)

- 平衡樹
  - Balancing Tree，是一種資料儲存的結構
  - 大部分關聯式資料庫管理系統，Index預設都是使用平衡樹結構
  - 若無Index，查詢時須掃描整個Table，稱為Full Table Scan
  - 若有Index，假設平衡樹有10層，查詢在10次比對之內就可結束
  - 對查詢(select)來說，加Index是一個用空間換時間的做法
  - 平衡樹結構占用大量空間，且每次資料異動(insert/delete/update)時都得重建，所以並非建越多越好

Index-介紹 (3/3)

- Index種類
  - 一般索引: 僅用來加速查詢，無其他特性。建立外來鍵(Foreign Key)時，就會自動建立一般索引
  - 唯一索引: 對應的欄位之值不可重複。建立唯一鍵(Unique Key)時，就會自動建立唯一索引 *註: MySQL中的唯一索引等效於唯一鍵
  - 主要索引: 對應的欄位之值不可重複、不能是null，且只能有一個。建立主要鍵(Primary Key)時，就會自動建立主要索引
  - 複合索引: 由2個以上欄位組成的索引，即是複合索引。上述提到的索引皆可以是複合索引

Index-新建Index (1/6)

- create index (1/2)
  - 語法

```
create [unique] index 索引名
on 資料表名(
    欄位名1[(長度)] [asc | desc],
    ..,
    欄位名N[(長度)] [asc | desc])
```

  - 說明
    - 索引名: 新建的Index名稱。命名習慣: IDX_資料表名_欄位名
    - 資料表名: 欲建立索引的Table名稱
    - 欄位名[(長度)] [asc | desc]: 組成索引的欄位。另可指定長度和排序方向

Index-新建Index (2/6)

- create index (2/2)
  - 範例

```
-- 範例-create index敘述-簡單Index
-- 新建一個Index，針對資料表EMP的欄位ENAME
create index IDX_EMP_ENAME
on EMP(ENAME);
```

Index-新建Index (3/6)

- create table with index (1/2)
  - 由於Index是針對Table，所以可在create table敘述新建Index
  - 語法

```
create table 資料表名(
    ..各欄位定義
    [unique] {index | key} 索引名(
        欄位名1[(長度)] [asc | desc], ..)
)
```

  - 說明
    - 索引名: 新建的Index名稱。命名習慣: IDX_資料表名_欄位名
    - 欄位名[(長度)] [asc | desc]: 組成索引的欄位。另可指定長度和排序方向

Index-新建Index (4/6)

- create table with index (2/2)
  - 範例

```
-- 範例-create table敘述-同時新建Index
-- 新建一個資料表DEPT，同時新建索引IDX_DEPT_DNAME
create table DEPT(
    DEPTNO int not null primary key,
    DNAME varchar(15),
    LOC varchar(15),
    unique index IDX_DEPT_DNAME(DNAME)
);
```

Index-新建Index (5/6)

- alter table with index (1/2)
  - 由於Index是針對Table，所以也可使用alter table敘述加入Index
  - 語法

```
alter table 資料表名
add [unique] {index | key} 索引名(
    欄位名1[(長度)] [asc | desc], ..)
```

  - 說明
    - 索引名: 新建的Index名稱。命名習慣: IDX_資料表名_欄位名
    - 欄位名[(長度)] [asc | desc]: 組成索引的欄位。另可指定長度和排序方向

Index-新建Index (6/6)

- alter table with index (2/2)
  - 範例

```
-- 範例-alter table敘述-加入Index
-- 替資料表DEPT，加入索引IDX_DEPT_LOC
alter table DEPT
add index IDX_DEPT_LOC(LOC);
```

Index-移除Index (1/2)

- 語法

```
drop index 索引名 on 資料表名
```

或

```
alter table 資料表名 drop index 索引名
```

- 說明
  - 索引名: 欲移除的Index名稱
  - 資料表名: Index所屬的Table名稱

Index-移除Index (2/2)

- 範例

```
-- 範例-drop index敘述
-- 移除名為IDX_DEPT_DNAME的Index
drop index IDX_DEPT_DNAME on DEPT;
```

```
-- 範例-alter table敘述-移除Index
-- 移除名為IDX_DEPT_LOC的Index
alter table DEPT drop index IDX_DEPT_LOC;
```

Index-適用時機

- 在 Index-介紹 (2/3) 提過，並非建越多越好，所以應考慮適用時機
- 資料筆數: 60萬+
- 敘述
  - 常執行select敘述的資料表
  - 少執行insert/delete/update敘述的資料表
- 值
  - 值域分布廣，重複少。反之，像性別只有2個值，就不適合
  - null少
- 欄位
  - 常出現在where、group by、order by子句
  - 常出現在join on後
  - 常使用在MAX()、MIN()

Index-Exercise12

01.請替資料表DEPT的欄位DNAME，新建唯一索引，並取出適當的Index名稱

02.請替資料表EMP的欄位(DEPTNO, ENAME)，新建複合索引，Index名稱為IDX_EMP_DEPTNO&ENAME

03.請新建一個資料表PERSON，欄位描述如下..

Column Name	DataType(Length)	not null?	PK?
ID	int	Y	Y
NAME	varchar(50)	Y	

並同時替欄位NAME加上一般索引

04.請使用alter table敘述，替資料表DEPT的欄位(DNAME, LOC)加入複合索引

05.請移除第04題建立的索引