1. Relational Database Development

152-156

Views, Indexes & Security

| Notes | Activity | |
| --- | --- | --- |
| 1. Quick Links & Text References    * [View Description](#viewdesc) Pages 182 – 183  187    * [Creating a View](#createview) Pages 183 – 187    * [Using a View](#usingview) Pages 184+    * [Renaming View Fields](#renamefields) Pages 185+    * [Views Based on Joins](#joinviews) Pages 186 – 187  189 – 192    * [Statistics Views](#statistics) Pages 186 – 187    * [Deleting a View](#delview) Page 192    * [Index Concepts](#indexconcepts) Pages 196 – 198    * [Creating an Index](#createindex) Page 199    * [Using an Index](#useindex) Pages    * [Displaying Current Indexes](#displayindex) Pages    * [Deleting an Index](#delindex) Page 200    * [Database Security](#security) Pages 193+    * [GRANT Table Privileges](#grant) Pages 193+    * [GRANT Field Privileges](#grantfield) Page 194    * [Restricting Record Access](#restrictrecord) Pages    * [REVOKE Privileges](#revoke) Page 195    * [Showing Privileges](#showpriv) Pages |  | |
| 1. View Description    * A view is simplistically a saved query.      + You designate which fields, and which records are to be included in the view using a SELECT statement      + This is sometimes done for security reasons. By only allowing users to see the data in a view you restrict their access to the full table/database.      + Views can also be used as reporting tools.      + Views can also be used to treat fields from multiple tables as if they are all in one table.      + Finally, views can be used to simplify complex queries. The view filters out certain fields and records, or calculates statistics. The view’s results can then be used in another query.    * Nothing is really saved except the description of what fields and records to include.      + Whenever the view is used, the data shown reflects the current contents of the table(s) used in the view.    * The book describes a view as a *window* to the data in the table—a window that restricts the data that is available. |  | |
| 1. Creating a View    * **CREATE VIEW** *viewname* **AS SELECT** *statement to select fields and records from another table*    * Executing this command creates a view—the results of the view are not shown.    * In the MySQL Workbench, views appear in the Navigator under Views (logically enough) (below Tables)      + Might need to refresh to see newly created views | Create a view *vwColomaEmployees* of all fields in tblemployees from plant 1 | |
| 1. Using a View    * Use the view name where you would normally use a table name in any query      + SQL will combine the query that created the view with the query that uses the view.   **CREATE VIEW** vwCsBooks **AS**  **SELECT** \* **FROM** tblbooks  **WHERE** bookType=’CS’  **SELECT** bookTitle, bookPrice  **FROM** vwCsBooks   1. WHERE bookPrice > 25  ORDER BY bookPrice DESC;    * Behind the scenes, SQL would combine the SELECT clause of the CREATE VIEW command with the SELECT statement that uses that view to form the following SELECT statement:   **SELECT** bookTitle, bookPrice  **FROM** tblBooks  **WHERE** bookPrice > 25  **AND** bookType= ‘CS’  **ORDER** **BY** bookPrice **DESC;**   * + This combination occurs behind the scenes, the user never knows it’s happening. | Display all records in view  Display all plant 1 union members  Discuss combined query what really occurs | |
| 1. Renaming Fields in a View    * When you create a view, you can change the names of the fields included in the view   **CREATE** **VIEW** vwCsBooks (title, price, type) **AS**  **SELECT** bookTitle, bookPrice, bookType  **FROM** tblBooks   1. WHERE bookType = ‘CS’;    * Note the **SELECT** clause must use the original field name in the **WHERE** clause    * When using statistics (aggregate) functions in the **SELECT** clause, you must specify a new name for every field in the view. You cannot rename the fields in the **SELECT** clause.   **CREATE** **VIEW** vwBookTotal (type, total) **AS**  **SELECT** bookType, **SUM**(bookPrice) **FROM** tblBooks   1. GROUP BY bookType;    * Because the fields have been renamed for the view, you must use the new field names when *using* the view. 2. SELECT title, price FROM vwCsBooks; | Create a view of all February trips,  tripDate, city, days  Show February trips to Chicago sorted by date | |
| 1. Views Based on Joins    * You can easily combine the fields (and rename them if you wish) from multiple tables to form a view.      + In the **SELECT** clause, join the tables.      + From the View user’s perspective, all the fields are now part of one *virtual table*.    * When you create these views, you have to be very careful about allowing users to insert, update and delete records from a view.      + Views often do not contain all the fields from one table. If you **INSERT** a new record, what values should be inserted in the fields that are not part of the view?      + If you **DELETE** a record from a view based on a join, should the records from both tables be deleted?      + It’s usually much safer to only allow the user to look at, but not modify, the data in a view (see security discussion below). But, that may not always meet the project requirements. Think about the ramifications of modifying data in a view. | Create a view of all employee information combined with all plant information for plant 1. | |
| 1. Views Based on Statistics    * Views can also be created that are based on queries that generate statistics (SUM, AVG, COUNT, etc)   **CREATE** **VIEW** vwBooksPerBranch (branch, totalInv) **AS**  **SELECT** branchName, SUM(unitsOnHand) **FROM** tblBranches **INNER JOIN** tblInventory  **USING** (branchNumber)   1. GROUP BY branchName;    * This view will show the total number of books in each branch. This is a good example of a view used as a reporting tool. The view can be used as the basis of another query as well.    * Views based on statistics NEVER have update, delete, or insert rights.    * See Renaming Fields in a View above for renaming rules that apply to views with statistics | Create view that displays the number of union and nonunion members at each plant.  Try some selects. | |
| 1. Deleting a View    * **DROP VIEW** *viewname*      + Note, views cannot be replaced by running another **CREATE** **VIEW** command. You must drop the view before creating the new version. |  | |
| 1. Index Concepts    * By default, database tables are sorted by primary key      + Especially in tables with autonumber fields, this is rarely useful    * Indexes store pointers to the original records in a different sort order      + This makes displaying the records in that order quicker      + Also significantly improves searching      + Also improves joins (create index for foreign key)      + With today’s powerful computers, this speed benefit is only realized on *large* tables    * Indexes can also be built on a combination of fields (similar to multi-field sorts) |  | |
| * + There are disadvantages to creating indexes     - Whenever data is added or changed, indexes need to be updated (takes time)     - Indexes take up additional storage space     - Have to balance the time lost adding/updating with the time gained when querying |  | |
| 1. Creating an Index   **CREATE INDEX** *idxName* **ON** *tblName* (*fldName1, fldName2* **DESC**);   * + Indexes can be created on one or more fields     - Fields must be designated **NOT NULL**   + If appropriate, the field can be sorted in descending order   + In MySQL Workbench, indexes are shown in the Navigator under the appropriate table  🡨 | Create an index for tblemployees based on lastname, firstname. | |
| * + Indexes can also be created when you define the table **CREATE TABLE** *tblName**field lists* **INDEX**(*field1, field2*); |  | |
| 1. Using an Index    * Indexes are **automatically** used whenever your query includes a **WHERE** clause that includes the index fields **in the same order they are specified in the index** or when your query includes an **ORDER BY** clause that includes the index fields in the same order they are specified in the index. | Select lastname, firstname from employees. Note: sorted.  Select just last. Just first (not sorted) | |
| 1. Displaying Current Indexes    * **SHOW INDEXES FROM** *tblName***;** | Try it | |
| 1. Deleting an Index    * **DROP INDEX** *idxName* **ON** *tblName***;** |  | |
| 1. Database Security    * SQL databases provide two levels of security.      + Views provide some level of security by restricting what data a user can see.      + The **GRANT** and **REVOKE** commands provide varying levels of access to tables    * The types of privileges that are available in SQL vary depending on the implementation. MySQL allows you to control the following privileges:      + Select      + Insert      + Update      + Delete      + Create      + Drop      + Grant      + References      + Index      + Alter | Because we are using a portable version of MySQL with only one user, these commands cannot be demonstrated. | |
| * + **GRANT** *privileges* **ON** *tblName* **TO** *usernames*;     - *privileges* can include many privileges, separated by commas     - *username* can include many users, separated by commas     - Only one *tablename* may be specified in a **GRANT** statement (despite what the book says)     - *usernames* can also be the keyword **PUBLIC**, granting rights to everyone     - *privileges* can also be the keyword **ALL**, granting all rights to the specified user, EXCEPT the Grant privilege. |  | |
| * + **GRANT** **SELECT** (*fieldlist)* **ON** *tablename* **TO** *usernames;*   + **GRANT** **UPDATE** (*fieldlist)* **ON** *tablename* **TO** *usernames;*     - When granting **SELECT** or **UPDATE** privileges, you can restrict the fields the users can view or change     - Note the field list follows the privilege, NOT the table name as shown in the book     - Field lists cannot be specified for other privileges     - If you don’t specify the field list, all fields are included | |  |
| * + **GRANT** *privileges* **ON** *tablename* **TO** *usernames* **WITH GRANT OPTION**;     - The WITH GRANT OPTION clause allows the users specified to pass their privileges to other users       * Only the privileges they have been given. A user who has not been given **DELETE** privileges cannot grant that privilege to someone else.       * When privileges are revoked, from a user, the privilege is also revoked from anyone the user granted the privilege to.   + Example: Let’s say I have **DELETE** privileges **WITH GRANT OPTION** and grant **DELETE** privileges to you. If the **DELETE** privilege is revoked from me, you lose **DELETE** privileges too. |  | |
| 1. Restricting Record Access    * In some instances, you’ll want to restrict the users’ access to records in addition to restricting column and table access.    * First build a view that includes only the columns and records you want the users to be able to access    * Then, grant privileges to the view.    * In these notes, wherever you see *tablename*, you can insert *viewname* instead.    * Example: Let’s say user THOMPSON is your representative to the New York publishers. THOMPSON should be able to see, change and delete publisher data, but only for those publishers he has responsibility for. |  | |
| **CREATE** **VIEW** vwNyPubs **AS**  **SELECT** \* **FROM** tblPublishers  **WHERE** publisherState=’NY’;  **GRANT** **SELECT**, **UPDATE**, **DELETE** **ON** vwNyPubs   1. TO Thompson; |  | |
| * + **REVOKE** *privileges* **ON** *tablename* **FROM** *usernames;*     - When you want to take privileges away from users, you use the **REVOKE** command.     - The **REVOKE** command is structured just like the **GRANT** command       * Can specify multiple privileges and users, but only one table       * Can use **PUBLIC** and **ALL**       * **WITH** **GRANT** **OPTION** doesn’t apply     - Examples:   **REVOKE** **ALL** **ON** tblBooks **FROM** **PUBLIC**;  **REVOKE** **UPDATE**, **DELETE** **ON** tblBooks **FROM** Davis, Johnson; |  | |
| 1. Showing Privileges    * **SHOW GRANTS**;    * **SHOW GRANTS FOR** *username*; |  | |