**Assignment 8: Views**

Here's the schema of the Movie-Rating database:   
  
**Movie** ( mID, title, year, director )   
English: There is a movie with ID number *mID*, a *title*, a release *year*, and a *director*.   
  
**Reviewer** ( rID, name )   
English: The reviewer with ID number *rID* has a certain *name*.   
  
**Rating** ( rID, mID, stars, ratingDate )   
English: The reviewer *rID* gave the movie *mID* a number of *stars* rating (1-5) on a certain *ratingDate*.   
  
In addition to the base tables, three views are also created:   
  
View **LateRating** contains movie ratings after January 20, 2011. The view contains the movie ID, movie title, number of stars, and rating date.   
  
create view LateRating as   
select distinct R.mID, title, stars, ratingDate   
from Rating R, Movie M   
where R.mID = M.mID   
and ratingDate > '2011-01-20'   
  
View **HighlyRated** contains movies with at least one rating above 3 stars. The view contains the movie ID and movie title.   
  
create view HighlyRated as   
select mID, title   
from Movie   
where mID in (select mID from Rating where stars > 3)   
  
View **NoRating** contains movies with no ratings in the database. The view contains the movie ID and movie title.   
  
create view NoRating as   
select mID, title   
from Movie   
where mID not in (select mID from Rating)   
  
Your exercises will run over a small data set conforming to the schema, with the views predefined. The schema and data is in viewMovie.sql file.

1. Write an instead-of trigger that enables updates to the title attribute of view **LateRating**.   
     
   **Policy:** Updates to attribute title in LateRating should update Movie.title for the corresponding movie. (You may assume attribute mID is a key for table Movie.) Make sure the mID attribute of view LateRating has not also been updated -- if it has been updated, don't make any changes. Don't worry about updates to stars or ratingDate.

Check:

First run the following data modification statement(s): *update LateRating set title = 'Late Favorite' where stars > 2; update LateRating set mID = 100, title = 'Don't change'*.

Then query the view: *select \* from LateRating*

*View Result:*

|  |  |  |  |
| --- | --- | --- | --- |
| 101 | Late Favorite | 2 | 2011-01-22 |
| 101 | Late Favorite | 4 | 2011-01-27 |
| 103 | Late Favorite | 3 | 2011-01-27 |
| 104 | E.T. | 2 | 2011-01-22 |
| 108 | Raiders of the Lost Ark | 2 | 2011-01-30 |

*T*hen run the following query: *select \* from Movie order by mID*

*Your Query Result:*

|  |  |  |  |
| --- | --- | --- | --- |
| 101 | Late Favorite | 1939 | Victor Fleming |
| 102 | Star Wars | 1977 | George Lucas |
| 103 | Late Favorite | 1965 | Robert Wise |
| 104 | E.T. | 1982 | Steven Spielberg |
| 105 | Titanic | 1997 | James Cameron |
| 106 | Snow White | 1937 | <NULL> |
| 107 | Avatar | 2009 | James Cameron |
| 108 | Raiders of the Lost Ark | 1981 | Steven Spielberg |

1. Write an instead-of trigger that enables updates to the stars attribute of view **LateRating**.   
     
   **Policy:** Updates to attribute stars in LateRating should update Rating.stars for the corresponding movie rating. (You may assume attributes [mID,ratingDate] together are a key for table Rating.) Make sure the mID and ratingDate attributes of view LateRating have not also been updated -- if either one has been updated, don't make any changes. Don't worry about updates to title.

Check:

First run the following data modification statement(s):

*update LateRating set stars = stars - 2 where stars > 2;*

*update LateRating set mID = 100, stars = stars + 2; update LateRating set ratingDate = null, stars = stars + 2.*

Then query the view: *select \* from LateRating*

View Result:

|  |  |  |  |
| --- | --- | --- | --- |
| 101 | Gone with the Wind | 2 | 2011-01-22 |
| 101 | Gone with the Wind | 2 | 2011-01-27 |
| 103 | The Sound of Music | 1 | 2011-01-27 |
| 104 | E.T. | 2 | 2011-01-22 |
| 108 | Raiders of the Lost Ark | 2 | 2011-01-30 |

Then run the following query: *select \* from Rating order by mID, stars*  
Your Query Result:

|  |  |  |  |
| --- | --- | --- | --- |
| 201 | 101 | 2 | 2011-01-22 |
| 201 | 101 | 2 | 2011-01-27 |
| 202 | 106 | 4 | <NULL> |
| 203 | 103 | 2 | 2011-01-20 |
| 203 | 108 | 2 | 2011-01-30 |
| 203 | 108 | 4 | 2011-01-12 |
| 204 | 101 | 3 | 2011-01-09 |
| 205 | 103 | 1 | 2011-01-27 |
| 205 | 104 | 2 | 2011-01-22 |
| 205 | 108 | 4 | <NULL> |
| 206 | 106 | 5 | 2011-01-19 |
| 206 | 107 | 3 | 2011-01-15 |
| 207 | 107 | 5 | 2011-01-20 |
| 208 | 104 | 3 | 2011-01-02 |

1. Write an instead-of trigger that enables updates to the mID attribute of view **LateRating**.   
     
   **Policy:** Updates to attribute mID in LateRating should update Movie.mID and Rating.mID for the corresponding movie. Update all Rating tuples with the old mID, not just the ones contributing to the view. Don't worry about updates to title, stars, or ratingDate.

To check your trigger, first run the following data modification statement(s): *update LateRating set mID = mID+50 where stars = 2*.   
  
Then query the view: *select \* from LateRating*  
View Result:

|  |  |  |  |
| --- | --- | --- | --- |
| 103 | The Sound of Music | 3 | 2011-01-27 |
| 151 | Gone with the Wind | 2 | 2011-01-22 |
| 151 | Gone with the Wind | 4 | 2011-01-27 |
| 154 | E.T. | 2 | 2011-01-22 |
| 158 | Raiders of the Lost Ark | 2 | 2011-01-30 |

Then run the following query: *select M.mID, title, stars from Movie M, Rating R where M.mID = R.mID order by M.mID, stars*  
Your Query Result:

|  |  |  |
| --- | --- | --- |
| 103 | The Sound of Music | 2 |
| 103 | The Sound of Music | 3 |
| 106 | Snow White | 4 |
| 106 | Snow White | 5 |
| 107 | Avatar | 3 |
| 107 | Avatar | 5 |
| 151 | Gone with the Wind | 2 |
| 151 | Gone with the Wind | 3 |
| 151 | Gone with the Wind | 4 |
| 154 | E.T. | 2 |
| 154 | E.T. | 3 |
| 158 | Raiders of the Lost Ark | 2 |
| 158 | Raiders of the Lost Ark | 4 |
| 158 | Raiders of the Lost Ark | 4 |

1. Write an instead-of trigger that enables deletions from view **HighlyRated**.   
     
   **Policy:** Deletions from view HighlyRated should delete all ratings for the corresponding movie that have stars > 3.

To check your trigger, first run the following data modification statement(s): *delete from HighlyRated where mID > 106*.   
  
Then query the view: *select \* from HighlyRated*

View Result:

|  |  |
| --- | --- |
| 101 | Gone with the Wind |
| 106 | Snow White |

Then run the following query: *select \* from Rating order by mID desc*  
Your Query Result:

|  |  |  |  |
| --- | --- | --- | --- |
| 201 | 101 | 2 | 2011-01-22 |
| 201 | 101 | 4 | 2011-01-27 |
| 202 | 106 | 4 | <NULL> |
| 203 | 103 | 2 | 2011-01-20 |
| 203 | 108 | 2 | 2011-01-30 |
| 204 | 101 | 3 | 2011-01-09 |
| 205 | 103 | 3 | 2011-01-27 |
| 205 | 104 | 2 | 2011-01-22 |
| 206 | 106 | 5 | 2011-01-19 |
| 206 | 107 | 3 | 2011-01-15 |
| 208 | 104 | 3 | 2011-01-02 |

1. Write an instead-of trigger that enables deletions from view **HighlyRated**.   
     
   **Policy:** Deletions from view HighlyRated should update all ratings for the corresponding movie that have stars > 3 so they have stars = 3.

To check your trigger, first run the following data modification statement(s): *delete from HighlyRated where mID > 106*.   
  
Then query the view: *select \* from HighlyRated*

View Result:

|  |  |
| --- | --- |
| 101 | Gone with the Wind |
| 106 | Snow White |

Then run the following query: *select \* from Rating order by mID desc*  
Your Query Result:

|  |  |  |  |
| --- | --- | --- | --- |
| 201 | 101 | 2 | 2011-01-22 |
| 201 | 101 | 4 | 2011-01-27 |
| 202 | 106 | 4 | <NULL> |
| 203 | 103 | 2 | 2011-01-20 |
| 203 | 108 | 2 | 2011-01-30 |
| 203 | 108 | 3 | 2011-01-12 |
| 204 | 101 | 3 | 2011-01-09 |
| 205 | 103 | 3 | 2011-01-27 |
| 205 | 104 | 2 | 2011-01-22 |
| 205 | 108 | 3 | <NULL> |
| 206 | 106 | 5 | 2011-01-19 |
| 206 | 107 | 3 | 2011-01-15 |
| 207 | 107 | 3 | 2011-01-20 |
| 208 | 104 | 3 | 2011-01-02 |

1. Write an instead-of trigger that enables insertions into view **HighlyRated**.   
     
   **Policy:** An insertion should be accepted only when the (mID,title) pair already exists in the Movie table. (Otherwise, do nothing.) Insertions into view HighlyRated should add a new rating for the inserted movie with rID = 201, stars = 5, and NULL ratingDate.

To check your trigger, first run the following data modification statement(s): *insert into HighlyRated values (104, 'E.T.'); insert into HighlyRated values (105, 'Titanic 2')*.   
  
Then query the view: *select \* from HighlyRated*  
View Result:

|  |  |
| --- | --- |
| 101 | Gone with the Wind |
| 104 | E.T. |
| 106 | Snow White |
| 107 | Avatar |
| 108 | Raiders of the Lost Ark |

Then run the following query: *select \* from Rating order by stars desc, mID*  
Your Query Result:

|  |  |  |  |
| --- | --- | --- | --- |
| 201 | 101 | 2 | 2011-01-22 |
| 201 | 101 | 4 | 2011-01-27 |
| 201 | 104 | 5 | <NULL> |
| 202 | 106 | 4 | <NULL> |
| 203 | 103 | 2 | 2011-01-20 |
| 203 | 108 | 2 | 2011-01-30 |
| 203 | 108 | 4 | 2011-01-12 |
| 204 | 101 | 3 | 2011-01-09 |
| 205 | 103 | 3 | 2011-01-27 |
| 205 | 104 | 2 | 2011-01-22 |
| 205 | 108 | 4 | <NULL> |
| 206 | 106 | 5 | 2011-01-19 |
| 206 | 107 | 3 | 2011-01-15 |
| 207 | 107 | 5 | 2011-01-20 |
| 208 | 104 | 3 | 2011-01-02 |

1. Write an instead-of trigger that enables insertions into view **NoRating**.   
     
   **Policy:** An insertion should be accepted only when the (mID,title) pair already exists in the Movie table. (Otherwise, do nothing.) Insertions into view NoRating should delete all ratings for the corresponding movie.

To check your trigger, first run the following data modification statement(s): *insert into NoRating values (104, 'E.T.'); insert into NoRating values (110, 'Avatar')*.   
  
Then query the view: *select \* from NoRating*

View Result:

|  |  |
| --- | --- |
| 102 | Star Wars |
| 104 | E.T. |
| 105 | Titanic |

Then run the following query: *select \* from Rating order by mID*  
Your Query Result:

|  |  |  |  |
| --- | --- | --- | --- |
| 201 | 101 | 2 | 2011-01-22 |
| 201 | 101 | 4 | 2011-01-27 |
| 202 | 106 | 4 | <NULL> |
| 203 | 103 | 2 | 2011-01-20 |
| 203 | 108 | 2 | 2011-01-30 |
| 203 | 108 | 4 | 2011-01-12 |
| 204 | 101 | 3 | 2011-01-09 |
| 205 | 103 | 3 | 2011-01-27 |
| 205 | 108 | 4 | <NULL> |
| 206 | 106 | 5 | 2011-01-19 |
| 206 | 107 | 3 | 2011-01-15 |
| 207 | 107 | 5 | 2011-01-20 |

1. Write an instead-of trigger that enables deletions from view **NoRating**.   
     
   **Policy:** Deletions from view NoRating should delete the corresponding movie from the Movie table.

To check your trigger, run the following data modification statement(s): *delete from NoRating where title = 'Titanic'*.   
  
Then query the view: *select \* from NoRating*

View Result:

|  |  |
| --- | --- |
| 102 | Star Wars |

Then run the following query: *select \* from Movie order by title*  
Your Query Result:

|  |  |  |  |
| --- | --- | --- | --- |
| 101 | Gone with the Wind | 1939 | Victor Fleming |
| 102 | Star Wars | 1977 | George Lucas |
| 103 | The Sound of Music | 1965 | Robert Wise |
| 104 | E.T. | 1982 | Steven Spielberg |
| 106 | Snow White | 1937 | <NULL> |
| 107 | Avatar | 2009 | James Cameron |
| 108 | Raiders of the Lost Ark | 1981 | Steven Spielberg |

1. Write an instead-of trigger that enables deletions from view **NoRating**.   
     
   **Policy:** Deletions from view NoRating should add a new rating for the deleted movie with rID = 201, stars = 1, and NULL ratingDate.

To check your trigger, run the following data modification statement(s): *delete from NoRating*.   
  
Then query the view: *select \* from NoRating*  
View Result: *Empty result*  
  
Then run the following query: *select \* from Rating order by stars*  
Your Query Result:

|  |  |  |  |
| --- | --- | --- | --- |
| 201 | 101 | 2 | 2011-01-22 |
| 201 | 101 | 4 | 2011-01-27 |
| 201 | 102 | 1 | <NULL> |
| 201 | 105 | 1 | <NULL> |
| 202 | 106 | 4 | <NULL> |
| 203 | 103 | 2 | 2011-01-20 |
| 203 | 108 | 2 | 2011-01-30 |
| 203 | 108 | 4 | 2011-01-12 |
| 204 | 101 | 3 | 2011-01-09 |
| 205 | 103 | 3 | 2011-01-27 |
| 205 | 104 | 2 | 2011-01-22 |
| 205 | 108 | 4 | <NULL> |
| 206 | 106 | 5 | 2011-01-19 |
| 206 | 107 | 3 | 2011-01-15 |
| 207 | 107 | 5 | 2011-01-20 |
| 208 | 104 | 3 | 2011-01-02 |