CpSc 4620/6620: Database Management Systems (DBMS) (TEXNH Approach)

Relational Schema and SQL Queries
James Wang

Create Table

- Create the Account table:
  ```sql
  CREATE TABLE ACCOUNT (
  type INTEGER,
  username varchar(100),
  password varchar(100),
  email varchar(2048) primary key
  );
  ```
- Is it good to use email as the primary key for ACCOUNT?
  - Email sometimes is very long.
  - String comparison is more expensive.
  - Be better create an artificial integer key.
  - Some DBMS systems do not allow long key.

The use of artificial key

- Create the Account table:
  ```sql
  CREATE TABLE ACCOUNT (
  AccountID INTEGER AUTO_INCREMENT PRIMARY KEY ,
  email VARCHAR(1024) NOT NULL,
  username VARCHAR(100) NOT NULL,
  password VARCHAR(100) NOT NULL,
  type TINYINT NOT NULL
  );
  ```
- Create the Media table:
  ```sql
  CREATE TABLE MEDIA (
  MediaID INTEGER AUTO_INCREMENT PRIMARY KEY,
  type INTEGER NOT NULL,
  name varchar(100) NOT NULL,
  path varchar(4096) NOT NULL,
  lastaccess DATETIME NOT NULL
  );
  ```

Create Associative Table

- It is easy to represent an M:N relationship in ER diagram.
- However, in the relational model, an associative relation must be used to represent the M:N relationship
- Create the Downloads table:
  ```sql
  CREATE TABLE DOWNLOADS (
  DownloadID INTEGER AUTO_INCREMENT PRIMARY KEY,
  AccountID INTEGER REFERENCES ACCOUNT(AccountID),
  MediaID INTEGER NOT NULL REFERENCES MEDIA(MediaID),
  ip varchar(128) NOT NULL,
  downloadtime DATETIME NOT NULL,
  FOREIGN KEY(AccountID) REFERENCES ACCOUNT(AccountID),
  FOREIGN KEY(MediaID) REFERENCES MEDIA(MediaID)
  );
  ```

Deal with 1:N relationship

- We don’t need to create a table for uploadedby relationship because it is a 1:N relationship.
- In the relational model, relationships are represented through foreign keys.
  ```sql
  ALTER TABLE MEDIA ADD COLUMN AccountID INTEGER NOT NULL;
  ALTER TABLE MEDIA ADD COLUMN uploadtime DATETIME;
  ALTER TABLE MEDIA ADD FOREIGN KEY(AccountID) REFERENCES ACCOUNT(AccountID);
  ```
A Improved Version of ER Diagram

SQL Queries
- List all media files in the MeTube system:
  SELECT * FROM MEDIA WHERE 1;
- Show media name and path that were uploaded before 09/20/2007:
  SELECT name, path FROM MEDIA
  WHERE uploadTime < '2007-09-20';
- List the image files (type = 1) uploaded by User #1:
  SELECT * FROM MEDIA
  WHERE type = 1 AND AccountID = 1;

SQL Queries (more)
- List all media files downloaded by user "xyz" in the MeTube system:
  SELECT username, name, path, ip
  FROM ACCOUNT, MEDIA, DOWNLOADS
  WHERE username = 'xyz' AND
  ACCOUNT.AccountID = DOWNLOADS.AccountID AND
  DOWNLOADS.MediaID = MEDIA.MediaID;
- Find the name of the most recently downloaded media.
  SELECT name, downloadedtime
  FROM MEDIA, DOWNLOADS
  WHERE MEDIA.MediaID = DOWNLOADS.MediaID
  ORDER BY downloadedtime DESC
  LIMIT 1;

Syntax of SQL Statements
- Selection-List is in the form of T1.attrib, ..., T2.attrib, .... They are the expected attributes for the returned results.
- From-List is a list of tables separated by comma.
- DISTINCT: Eliminate the redundant rows in the return results.
- Qualification: Conditions for a record to be selected.

Expressions in SQL
- You can apply operations on the scalars (int, float, string, datetime, etc.) in the qualification or in the select-list.
  SELECT UPPER(name), DATEDIFF(CURRENT_DATE, downloadtime)
  FROM MEDIA, DOWNLOADS
  WHERE MEDIA.MediaID = DOWNLOADS.MediaID AND
  DATEDIFF(CURRENT_DATE, downloadtime) > 2;
- Note on strings:
  - Fixed (CHAR(x)) or variable length (VARCHAR(x))
  - Use single quotes: 'A string'
  - Special comparison operator: LIKE
  - Not equal: <>
- Typecasting: CAST(S.sid AS VARCHAR(255))

Operations on Two Queries
- Set operations can be applied on two queries.
  (SELECT ... FROM ... WHERE ...)
  (op)
  (SELECT ... FROM ... WHERE ...)
- (op): UNION, INTERSECT, MINUS/EXCEPT.
- Note: Not all DBMS support "MINUS/EXCEPT".
- Nested Queries:
  - IN, NOT IN
  - EXISTS, NOT EXISTS
- Where to use Nested Queries?
  - Anywhere in from-list or qualification.
Examples of Nested Queries

- Find users downloaded media 1 but not media 3:
  select username from account, downloads
  where account.accountid = downloads.accountid and
  mediaid = 1 and username not in (select username from account,
  downloads where account.accountid = downloads.accountid and
  mediaid = 3);

- Can we write a query by using EXISTS or NOT EXISTS to return the same results?

  how about nesting query in the from-list?
  select ua.username from
  ( select account.username,
  downloads.mediaid from account, downloads
  where account.accountid = downloads.accountid
  and mediaid = 1 ) as ua
  where ua.username not in ( select username
  from account, downloads where account.accountid =
  downloads.accountid and mediaid = 3 )

Aggregation

- GROUP BY
  select (group-attribs), (aggregate-operator)(attrib)
  from (relation) T1, (relation) T2, ...
  where (predicates)
  group by (group-list)

- Aggregate operators
  AVG, COUNT, SUM, MAX, MIN
  DISTINCT keyword for AVG, COUNT, SUM

- Example:
  find the numbers of times that users downloaded media 1 respectively.
  select accountid, count(accountid) as cnt
  from downloads
  where mediaid = 1
  group by accountid
  order by cnt

HAVING Clause

- What if you want to only show some groups?
  The HAVING clause lets you do a selection based on an aggregate (there must be 1 value per group):
  select accountid, count(accountid) as cnt
  from downloads
  where mediaid = 1
  group by accountid
  having cnt > 1

- Sometimes, we combine a nested GROUP BY query in the from-list to aggregate twice.

Beyond SELECT

- INSERT:
  insert into T(C1, C2, ...) values (V1, V2, ...)
  insert into T(C1, C2, ...) select ...

- DELETE:
  delete from tbl where condition-list

- UPDATE:
  update tbl
  set C1 = x1, C2 = x2
  where condition-list

JOIN

- INNER JOIN:
  regular JOIN will return results only when the join attribute matches in both tables involved.
  in MySQL JOIN == CROSS JOIN.

- LEFT JOIN:
  besides returning results matching the join attribute in both tables. It also returns results from left table that do not have matching rows in right table.
  some DBMS vendor use syntax LEFT OUTER JOIN.

- RIGHT JOIN:
  return results from right table that table that do not have matching rows in left table.
  some DBMS vendor use syntax RIGHT OUTER JOIN.

References