372 Group project

The Jaywalkers

Members:
Benya Chongolnee
Matthew Kerner
Jack Bartolone
Mathew Berry

Description of Application and Motivation:

The Jaywalker Music Company has assembled a database of their employees, artists, songs, albums, and sales information about the music. This database is very useful for the company in order to find ways to improve the company and make the lives of the customers and employees easier. This database will also be able to better focus on finding trends to personalize music sales. These will include looking at what customers like what kind of music, sales numbers on individual songs/artists, and things like the most popular and least popular genre of music. Creating a series of procedures, functions, and triggers to better understand the information in the database is very helpful for the company.

Motivation For Choosing:

The database was a positive learning experience because it offered a real world business scenario. The database is similar to the Eagle database since it offered tables such as Employee, Customer, Music, Album and other sales information. Because it had such a broad range of information to choose from, we were not limited to small scale questions. We could think how a real world music business might think. The group was able to think of creative examples using procedures, functions, and triggers in Oracle PL/SQL. Overall the group has gotten much more comfortable with PL/SQL and we see how powerful it is in a real world business scenario.

Description of Tables:

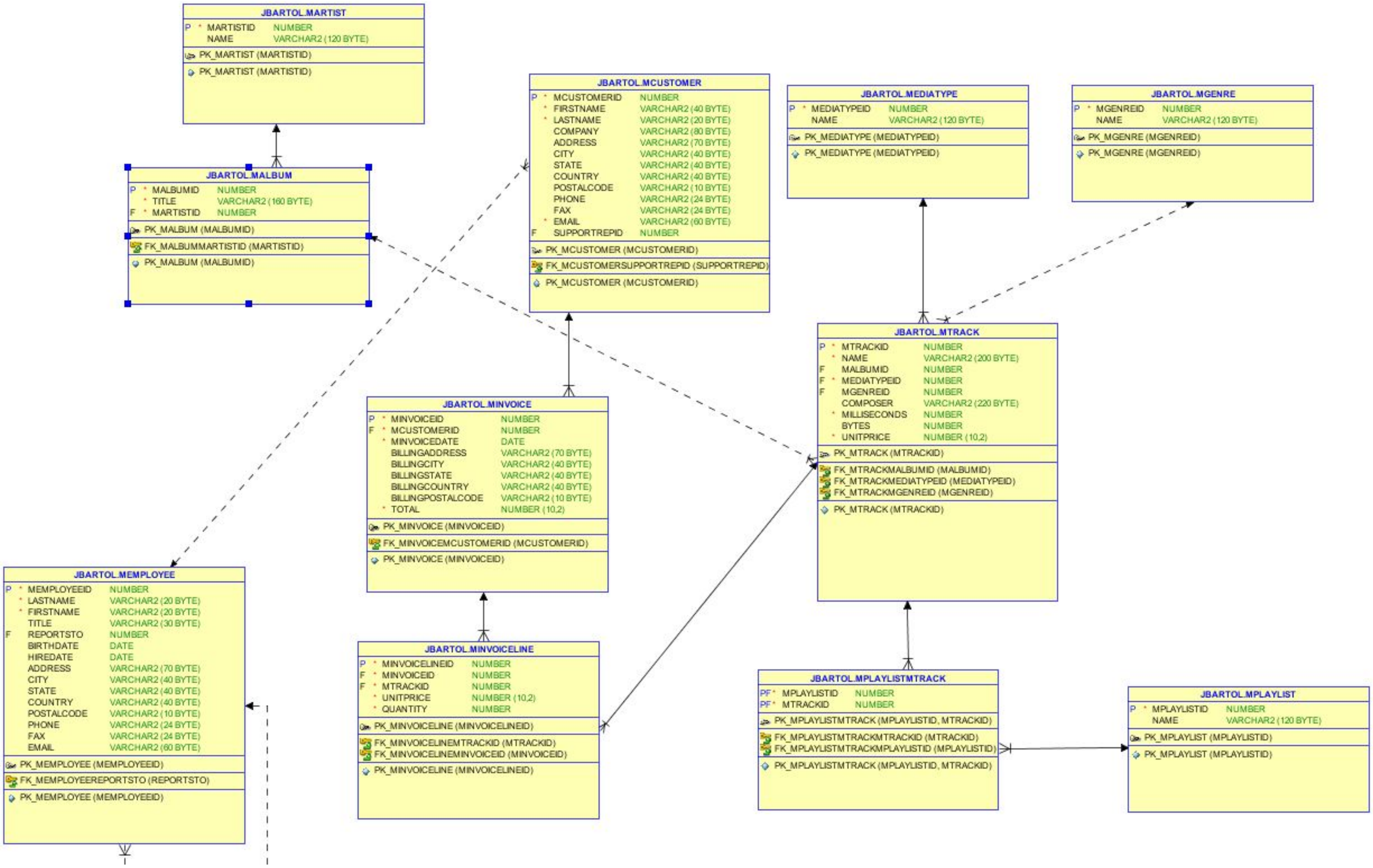
The database has an employees table, as well as a customer table. The bulk of the information in the database comes from tables related to music including album names, artist names, track names, genre, and other descriptions about the music. The database also keeps sales records for the music. The database contains 11 tables all populated with existing data. *A diagram of the database is attached in the next page.*

Description of Efforts:

Everyone in the group contributed evenly in various ways based on knowledge and SQL talents. Jack was very knowledgeable in exception handling which proved very valuable. Mathew was well rounded in the syntax of procedures and was a big help to Matt and Benya. Jack and Matt did most of the revised questions after the troubles with version 1 of Milestone 2. Mathew and Matt were mostly responsible for creating the powerpoint presentation for class, while Benya took charge of making and formatting this document to ensure that all of the requirements are being met. She also do the same for the powerpoint. Overall the team worked very well in a balanced way. We all attended the meetings as assigned, and we all contribute to the project to the best of our ability.

The only major obstacle was the mystery member of Connor Wright. We discovered he joined our group on Blackboard on 11/27/2017. We would like to submit for the record that he did not contribute in any way to this project. We tried to email him regarding this issue, and we have not heard back from him. None of the group members know who this person is.

Diagram:



```
JBARTOL.MARTIST
P * MARTISTID      NUMBER
    NAME           VARCHAR2 (120 BYTE)
PK_MARTIST (MARTISTID)
PK_MARTIST (MARTISTID)
```

```
JBARTOL.MALBUM
P * MALBUMID       NUMBER
  * TITLE          VARCHAR2 (160 BYTE)
F * MARTISTID      NUMBER
PK_MALBUM (MALBUMID)
FK_MALBUMMARTISTID (MARTISTID)
PK_MALBUM (MALBUMID)
```

```
JBARTOL.MCUSTOMER
P * MCUSTOMERID    NUMBER
  * FIRSTNAME      VARCHAR2 (40 BYTE)
  * LASTNAME       VARCHAR2 (20 BYTE)
    COMPANY        VARCHAR2 (80 BYTE)
    ADDRESS        VARCHAR2 (70 BYTE)
    CITY           VARCHAR2 (40 BYTE)
    STATE          VARCHAR2 (40 BYTE)
    COUNTRY        VARCHAR2 (40 BYTE)
    POSTALCODE     VARCHAR2 (10 BYTE)
    PHONE          VARCHAR2 (24 BYTE)
    FAX            VARCHAR2 (24 BYTE)
  * EMAIL          VARCHAR2 (60 BYTE)
F   SUPPORTREPID   NUMBER
PK_MCUSTOMER (MCUSTOMERID)
FK_MCUSTOMERSUPPORTREPID (SUPPORTREPID)
PK_MCUSTOMER (MCUSTOMERID)
```

```
JBARTOL.MEDIATYPE
P * MEDIATYPEID    NUMBER
    NAME           VARCHAR2 (120 BYTE)
PK_MEDIATYPE (MEDIATYPEID)
PK_MEDIATYPE (MEDIATYPEID)
```

```
JBARTOL.MGENRE
P * MGENREID       NUMBER
    NAME           VARCHAR2 (120 BYTE)
PK_MGENRE (MGENREID)
PK_MGENRE (MGENREID)
```

```
JBARTOL.MTRACK
P * MTRACKID       NUMBER
  * NAME           VARCHAR2 (200 BYTE)
F   MALBUMID       NUMBER
F * MEDIATYPEID    NUMBER
F   MGENREID       NUMBER
    COMPOSER       VARCHAR2 (220 BYTE)
  * MILLISECONDS   NUMBER
    BYTES          NUMBER
  * UNITPRICE      NUMBER (10,2)
PK_MTRACK (MTRACKID)
FK_MTRACKMALBUMID (MALBUMID)
FK_MTRACKMEDIATYPEID (MEDIATYPEID)
FK_MTRACKMGENREID (MGENREID)
PK_MTRACK (MTRACKID)
```

```
JBARTOL.MINVOICE
P * MINVOICEID          NUMBER
F * MCUSTOMERID         NUMBER
  * MINVOICEDATE        DATE
    BILLINGADDRESS      VARCHAR2 (70 BYTE)
    BILLINGCITY         VARCHAR2 (40 BYTE)
    BILLINGSTATE        VARCHAR2 (40 BYTE)
    BILLINGCOUNTRY      VARCHAR2 (40 BYTE)
    BILLINGPOSTALCODE   VARCHAR2 (10 BYTE)
  * TOTAL               NUMBER (10,2)
PK_MINVOICE (MINVOICEID)
FK_MINVOICEMCUSTOMERID (MCUSTOMERID)
PK_MINVOICE (MINVOICEID)
```

```
JBARTOL.MEMPLOYEE
P * MEMPLOYEEID    NUMBER
  * LASTNAME       VARCHAR2 (20 BYTE)
  * FIRSTNAME      VARCHAR2 (20 BYTE)
    TITLE          VARCHAR2 (30 BYTE)
F   REPORTSTO      NUMBER
    BIRTHDATE      DATE
    HIREDATE       DATE
    ADDRESS        VARCHAR2 (70 BYTE)
    CITY           VARCHAR2 (40 BYTE)
    STATE          VARCHAR2 (40 BYTE)
    COUNTRY        VARCHAR2 (40 BYTE)
    POSTALCODE     VARCHAR2 (10 BYTE)
    PHONE          VARCHAR2 (24 BYTE)
    FAX            VARCHAR2 (24 BYTE)
    EMAIL          VARCHAR2 (60 BYTE)
PK_MEMPLOYEE (MEMPLOYEEID)
FK_MEMPLOYEEREPORTSTO (REPORTSTO)
PK_MEMPLOYEE (MEMPLOYEEID)
```

```
JBARTOL.MINVOICELINE
P * MINVOICELINEID   NUMBER
F * MINVOICEID       NUMBER
F * MTRACKID         NUMBER
  * UNITPRICE        NUMBER (10,2)
  * QUANTITY         NUMBER
PK_MINVOICELINE (MINVOICELINEID)
FK_MINVOICELINEMTRACKID (MTRACKID)
FK_MINVOICELINEMINVOICEID (MINVOICEID)
PK_MINVOICELINE (MINVOICELINEID)
```

```
JBARTOL.MPLAYLISTMTRACK
PF* MPLAYLISTID    NUMBER
PF* MTRACKID       NUMBER
PK_MPLAYLISTMTRACK (MPLAYLISTID, MTRACKID)
FK_MPLAYLISTMTRACKMTRACKID (MTRACKID)
FK_MPLAYLISTMTRACKMPLAYLISTID (MPLAYLISTID)
PK_MPLAYLISTMTRACK (MPLAYLISTID, MTRACKID)
```

```
JBARTOL.MPLAYLIST
P * MPLAYLISTID    NUMBER
    NAME           VARCHAR2 (120 BYTE)
PK_MPLAYLIST (MPLAYLISTID)
PK_MPLAYLIST (MPLAYLISTID)
```

Questions & Answers:

/*1. How would you count the number of songs in a genre? */
/*The company may want to know how popular a certain genre is. Unpopular genres could then be consolidated or eliminated and the popular genre can be invested more.*/

```
CREATE OR REPLACE FUNCTION SongsnGenre(
                 given_genre jbartol.mgenre.mgenreID%type) RETURN number IS
                GENCOUNT NUMBER;
                invalid_num exception;
              BEGIN
                IF given_genre < 1 OR given_genre > 25 THEN raise invalid_num;
                END IF;
                SELECT COUNT(*) INTO gencount
                FROM jbartol.mtrack
                WHERE jbartol.mtrack.mgenreID = given_genre;
                RETURN GENCOUNT;
                exception
                when invalid_num then
                dbms_output.put_line('Invalid input. Please insert ID 1-25');
                return -1;
                when others then
                dbms_output.put_line('Error');
                return -1;
              END SongsnGenre;
              /
              select SongsnGenre(26) from dual;
```

/*2. How would you create a procedure that count the amount of customers from a given state?*/
/*By knowing the amount of customers from a given state, Jaywalker Music could concentrate advertising on it’s most popular states.*/

```
              CREATE OR REPLACE PROCEDURE statecustomer (
                 newstate   IN jbartol.mcustomer.state%TYPE
              ) AS
                 custnum   NUMBER;
                 negativecustid EXCEPTION;
              BEGIN
                 SELECT
                    COUNT(*)
```

```
    INTO
        custnum
    FROM
        jbartol.mcustomer
    WHERE
        jbartol.mcustomer.state = newstate;

    IF
        ( custnum = 0 )
    THEN
        RAISE negativecustid;
    END IF;
    dbms_output.put_line(custnum);
EXCEPTION
    WHEN negativecustid THEN
        dbms_output.put_line('No customers live in this state!');
END statecustomer;
```

/*3. How would you create a procedure to print the top three selling artists by quantity sold? */

/*Jaywalker Music could set a minimum sales threshold and any songs performing below this level could be cut from the library. This procedure allows them to get that data.*/

```
CREATE OR REPLACE PROCEDURE artisttop (
    given_amount INTEGER
) AS

    artistname   jbartol.mtrack.composer%TYPE;
    artistqty    jbartol.minvoiceline.quantity%TYPE;
    CURSOR best_artists IS SELECT
        composer,
        SUM(quantity)
    FROM
        jbartol.minvoiceline a
   INNER JOIN jbartol.mtrack b ON a.mtrackid = b.mtrackid
        WHERE
        composer IS NOT NULL
    GROUP BY
        composer
    ORDER BY
        SUM(quantity) DESC;
```

```
BEGIN
   OPEN best_artists;
   FETCH best_artists INTO artistname,artistqty;
   FOR i IN 1..given_amount LOOP
      dbms_output.put_line(artistname
      || ' '
      || artistqty);
      FETCH best_artists INTO artistname,artistqty;
   END LOOP;

END artisttop;
/
BEGIN
   artisttop(6);
END;
```

/* 4. How would you create a procedure to allow the renaming of state of the state of all employees?*/

/*The Jaywalker Music often uses tax loopholes in various states. To avoid lawsuits, they are constantly moving headquarters. This procedure would allow the company to easily change the home state of employees.*/

```
CREATE OR REPLACE PROCEDURE setstate (
   input1   IN jbartol.memployee.state%TYPE,
   input2   IN jbartol.memployee.state%TYPE
)
   IS
BEGIN
   UPDATE jbartol.memployee
      SET
         state = input1
   WHERE
      state = input2;

EXCEPTION
   WHEN storage_error THEN
      dbms_output.put_line('Could not adjust. Storage error or corruption has occurred');
   WHEN OTHERS THEN
      dbms_output.put_line('Other error');
END setstate;
/
Begin
```

```
        setState('AB','ZZ');
        End;
```

/* 5.How would you create a procedure to add a new employee first and last name?*/
/*Jaywalker Music is constantly hiring new employees. Having an easy way to populate the data would improve efficiency by speeding up the process.*/

```
        CREATE OR REPLACE PROCEDURE hireEmployee(fName IN
    JBARTOL.memployee.FirstName%type, lName IN
    JBARTOL.memployee.LastName%type) is
        Begin
        Insert into JBARTOL.memployee(memployeeID,Firstname,Lastname)
        VALUES ((select MAX(mEMPLOYEEID)+1 from
    JBARTOL.mEmployee),fName,lName);
        End;
        /
        Begin
        hireEmployee('Mathew','Berry');
        End;
```

/* 6. How would you create a procedure to remove all songs by a particular artist?*/
/*Artists are constantly leaving Jaywalker Music. This procedure makes it easy for the company to remove the artist, saving the company the embarrassment and storage space*/

```
        CREATE OR REPLACE PROCEDURE artistDelete(iArtistName in
    JBARTOL.mtrack.composer%Type) as
        Begin
        Delete
        from JBARTOL.mtrack where composer = iArtistName;
        exception
          when no_data_found then
            dbms_output.put_line('No artist found');
          when others then
            dbms_output.put_line('Other error');
        End;
        /
        Begin
        artistDelete('Billy Cobham');
        End;
```

/* 7. How would you create a procedure to search for customers, and display their first and last name is the customer is found*/

/*The billing department needs an easy way to search for customers to send debt collectors after them. They must ensure that the customer do exist in the database first.*/

```
CREATE OR REPLACE PROCEDURE customersearch (
   newcustid   IN jbartol.mcustomer.mcustomerid%TYPE
) AS

   firsttemp   jbartol.mcustomer.firstname%TYPE;
   lasttemp    jbartol.mcustomer.lastname%TYPE;
   CURSOR custsearch IS SELECT
      firstname,
      lastname
                 FROM
      jbartol.mcustomer
                 WHERE
      mcustomerid = newcustid;

BEGIN
   OPEN custsearch;
   FETCH custsearch INTO firsttemp,lasttemp;
   WHILE ( custsearch%found ) LOOP
      dbms_output.put_line(firsttemp
      || ' '
      || lasttemp);
      FETCH custsearch INTO firsttemp,lasttemp;
   END LOOP;

END customersearch;

EXECUTE customersearch('2');
```

/*8. How would you create a trigger for when an unauthorized user tries to insert, update, or delete from any of the tables?*/
/*The CEO’s nephew often hacks his laptop and tries to modify the database. The company needs an easy way to block this unauthorized access*/

```
          CREATE OR REPLACE TRIGGER unauthorized_alteration_mcustomer
BEFORE
           INSERT OR
           UPDATE OR
           DELETE ON jbartol.mcustomer DECLARE
          Allowed NUMBER;
           BEGIN
          IF inserting THEN
          /*Check if allowed*/
          SELECT COUNT(*)
          INTO allowed
          FROM user_tab_privs
          WHERE (grantee =
          (SELECT USER FROM dual
          )
          OR grantor =
          (SELECT USER FROM dual
          ))
          AND table_name = 'MCUSTOMER'
          AND privilege  = 'INSERT';
          IF Allowed       = 0 THEN
          raise_application_error(-20001, 'Unauthorized insertion on mcustomer
table');
          END IF;
          ELSIF updating THEN
          /*Check if allowed*/
          SELECT COUNT(*)
          INTO allowed
          FROM user_tab_privs
          WHERE (grantee =
          (SELECT USER FROM dual
          )
          OR grantor =
          (SELECT USER FROM dual
          ))
          AND table_name = 'MCUSTOMER'
          AND privilege  = 'UPDATE';
          IF Allowed       = 0 THEN
          raise_application_error(-20002, 'Unauthorized updating on mcustomer
table');
          END IF;
          ELSIF deleting THEN
          /*Check if allowed*/
          SELECT COUNT(*)
          INTO allowed
          FROM user_tab_privs
          WHERE (grantee =
          (SELECT USER FROM dual
          )
          OR grantor =
          (SELECT USER FROM dual
          ))
          AND table_name = 'MCUSTOMER'
          AND privilege  = 'DELETE';
          IF Allowed      = 0 THEN
          raise_application_error(-20003, 'Unauthorized deletion on mcustomer
table');
          END IF;
          END IF;
     END;
```

/* 9.How would one create a procedure to raise or lower the cost of all songs by a desired percentage? */

/*Sales are plummeting and the company needs an easy way to mark down song prices. In order to create more revenue, the company wants to put songs on sale to attract customer. This procedure would be the most efficient way to liquidate assets.*/

```
CREATE OR REPLACE PROCEDURE song_Price_Adjust(
        input_decimal IN number)
IS
 invalid_input exception;
BEGIN
  if not regexp_like(input_decimal,'^-?\d+(\.\d+)?$' )  then
   raise invalid_input;
  else
  UPDATE jbartol.mtrack SET unitprice = unitprice * (input_decimal/100);
  end if;
  exception
  when invalid_input then
  dbms_output.put_line('Invalid input. Please input a float value');
  when others then
  dbms_output.put_line('Unhandled error');
END song_Price_Adjust;
/
execute song_price_adjust(150);
/
select unitprice
from jbartol.mtrack;
```

/*10. How would you create a procedure to update customer names in the database?*/

/*Customers are often getting married and they sometimes request to get their last name changed. This procedure would make it easy for employees to keep the last names up to date*/

```
CREATE OR REPLACE FUNCTION update_name(
    originalname IN jbartol.mcustomer.lastname%type,
    newname      IN jbartol.mcustomer.lastname%type)
  return jbartol.mcustomer.lastname%type is
  reply jbartol.mcustomer.lastname%type;
BEGIN
     UPDATE jbartol.mcustomer
    SET lastname = newname
    where lastname = originalname;
    reply := 'Updated '|| originalname || ' to ' || newname;
    return reply;
    exception
          when no_data_found then
          return 'The name do not exist in the database';
        when others then
          return 'Other error';
END update_name;
/
begin
dbms_output.put_line(update_name('Helena', 'Helen'));
end;
```

Employee Package:

```
CREATE OR REPLACE PACKAGE pkg_employee
IS
  --hire employee
  PROCEDURE hireEmployee(
            fName IN JBARTOL.memployee.FirstName%type,
            lName IN JBARTOL.memployee.LastName%type);
  --change employee state
  PROCEDURE setState(
            input1 IN JBARTOL.memployee.state%type,
            input2 IN JBARTOL.memployee.state%type);
END pkg_employee;
/
CREATE OR REPLACE PACKAGE body pkg_employee
IS
  -- hire employee
  PROCEDURE hireEmployee(
            fName IN JBARTOL.memployee.FirstName%type,
            lName IN JBARTOL.memployee.LastName%type)
  IS
  BEGIN
            INSERT
            INTO JBARTOL.memployee
            (
            memployeeID,
            Firstname,
            Lastname
            )
            VALUES
            (
            (SELECT MAX(mEMPLOYEEID)+1 FROM JBARTOL.mEmployee
            )
            ,
            fName,
            lName
            );
  END;
--change employee state
  PROCEDURE setState
            (
            input1 IN JBARTOL.memployee.state%type,
            input2 IN JBARTOL.memployee.state%type
            )
  IS
  BEGIN
            UPDATE JBARTOL.MEMPLOYEE SET STATE = INPUT1 WHERE STATE =
INPUT2;
  END setState;
END pkg_employee;
```