# Report 1: Entity-Relationship Conceptual Modelling

The E-R components are:

**Music Band**: band-name, band webpage; each band has several releases. (1: M)

**Band Release**: title of release, year of release, release rate, release type.

**Songs:** songtitle, award; songs associate with band release, band produce songs.

**Musician**: musician name, personal website;

**Membership**: musician name, start year, end year, band name, role;

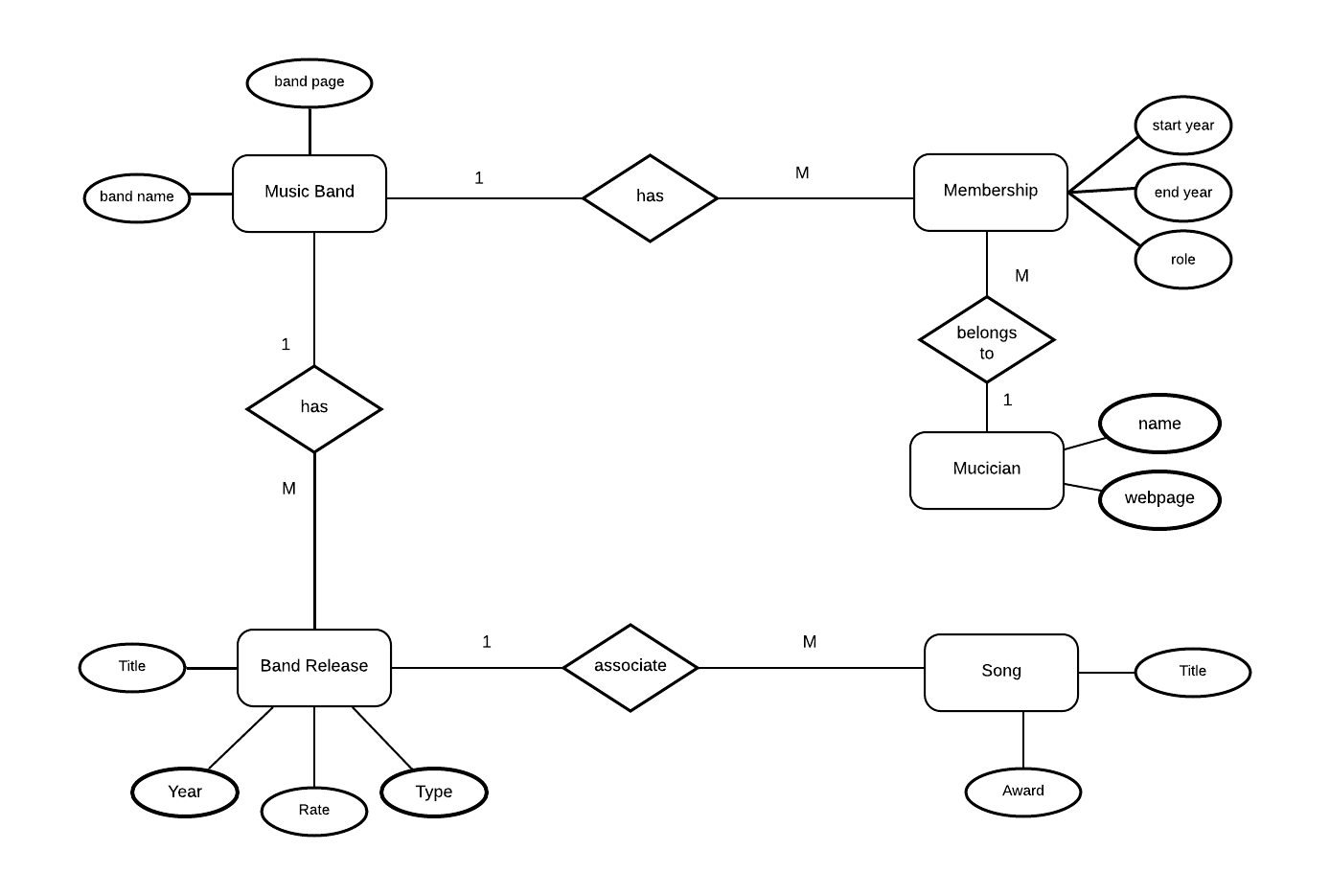


Figure 1: The E-R Diagram of “Band” Database

# Report 2: Relational Schema Modelling

## Relational Schema

**Music Band**: band-name, band webpage.

**Release**: Release-ID, release title, year of release, release rate, release type.

**Songs:** Release-ID，Songtitle, award.

**Musician**: Musician name, Musician webpage.

**Membership**: Membership-ID, Musician webpage, start year, end year, band name, Role.

It has been claimed that the different bands may have the same band name before it was registered as a trademark, as there are millions of bands online [1]. Therefore, this report assume that the “Band” is discriminated by a combination of band name and webpage. Although a musician for a specific time period is a member of a specific band only, the time period cannot be used as a primary key, for the primary key cannot be null value. A membership ID was therefore introduced as the primary key of membership.

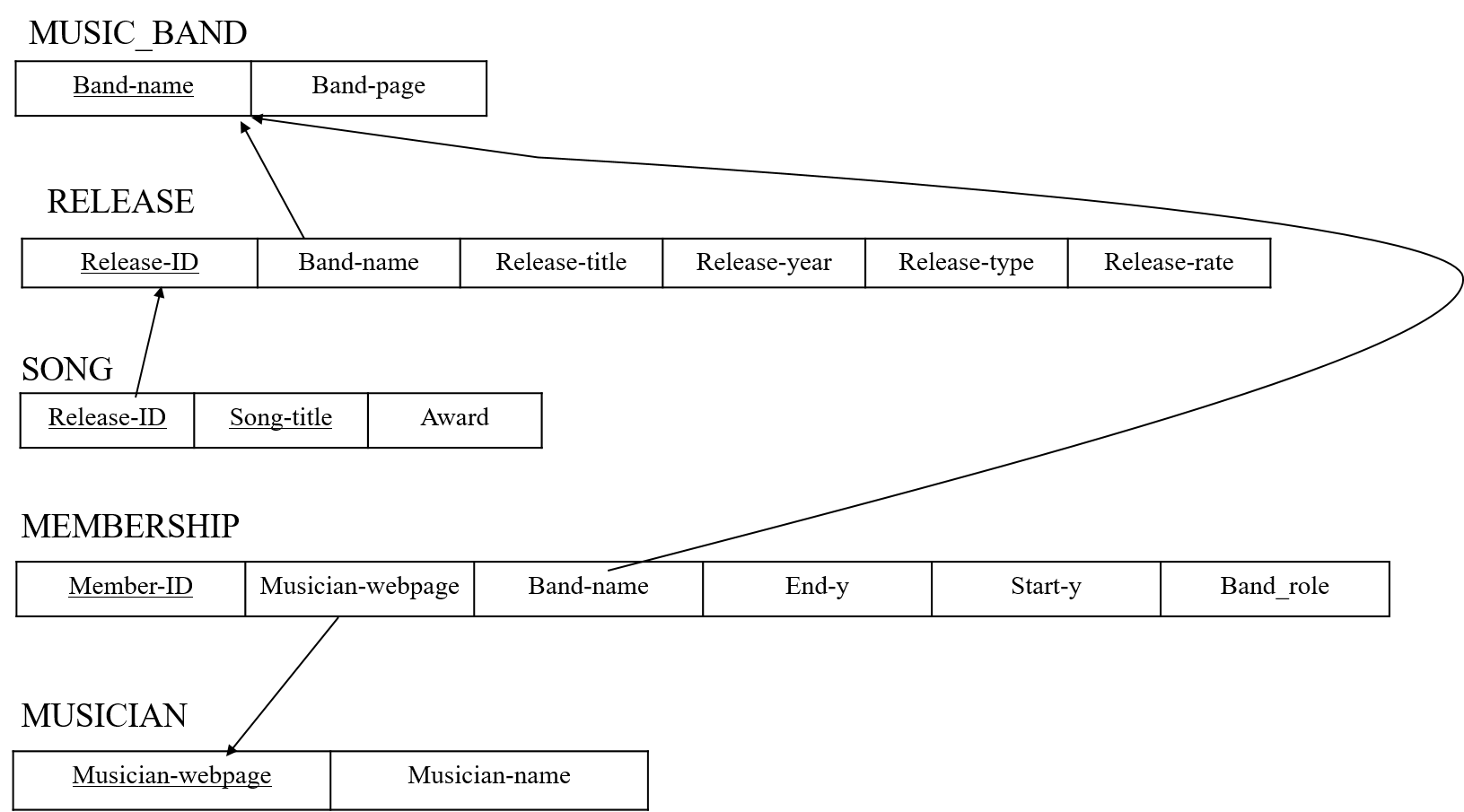


Figure 2: Relational Schema at BCNF

# Report 3.1: SQL CREATE

CREATE SCHEMA BANDS;

CREATE table MUSIC\_BAND

(Band\_name VARCHAR(32),

Band\_page VARCHAR(64),

PRIMARY KEY (Band\_name));

CREATE TABLE MUSICIAN

(Musician\_name VARCHAR(32),

Musician\_webpage VARCHAR(64),

PRIMARY KEY (Musician\_webpage));

CREATE TABLE RELEASE\_INFO

(Release\_ID CHAR(10),

Band\_name VARCHAR(32),

Release\_title VARCHAR(32),

Release\_year INT,

Release\_type VARCHAR(8),

Release\_rate INT,

PRIMARY KEY(Release\_ID),

FOREIGN KEY(Band\_name) REFERENCES MUSIC\_BAND(Band\_name)

ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE SONG

(RELEASE\_ID CHAR(10),

Song\_title CHAR(32),

Award BOOLEAN,

PRIMARY KEY (Release\_ID,Song\_title),

FOREIGN KEY(Release\_ID) REFERENCES RELEASE\_INFO(Release\_ID)

ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE MEMBERSHIP

(Member\_ID CHAR(10),

Musician\_webpage VARCHAR(64),

Band\_name VARCHAR(32),

End\_y INT,

Start\_y INT,

Band\_role VARCHAR(16),

PRIMARY KEY (Member\_ID),

FOREIGN KEY(Musician\_webpage) REFERENCES MUSICIAN(Musician\_webpage)

ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY(Band\_name) REFERENCES MUSIC\_BAND(Band\_name)

ON DELETE CASCADE ON UPDATE CASCADE);

Explanation of constrains:

# Report 3.2: SQL SELECT

SQL 1: Show the names of all current band members of the band “the CURE”.

SELECT Musician\_name

FROM MEMBERSHIP AS E, MUSICIAN AS U

WHERE Band\_name='The CURE' AND End\_y is NULL

AND E.Musician\_webpage = U.Musician\_webpage;

SQL 2: Find out if there exists a band with the same name as any member’s name of any band.

SELECT Band\_name, Musician\_name

FROM MUSIC\_BAND, MUSICIAN

WHERE Band\_name = Musician\_name;

SQL 3: Show the number of members in the database who have played “bass”

SELECT count(distinct musician\_webpage)

FROM MEMBERSHIP

WHERE band\_role = 'bass';

// what if two musician have a same name?

SQL 4: Show the discography of each drummer sorted by their name and year of release.

SELECT Release\_title, Musician\_name, Release\_year

FROM MEMBERSHIP as e, Release\_info as r,musician as u

WHERE (r.release\_year<=End\_y or End\_y is null)

AND (r.release\_year>=Start\_y OR Start\_y IS NULL)

AND e.musician\_webpage = u.musician\_webpage

AND Band\_role='drums'

AND e.band\_name = r.band\_name

ORDER BY Musician\_name, Release\_year;

# Report 4.1: Relational Algebra Tree

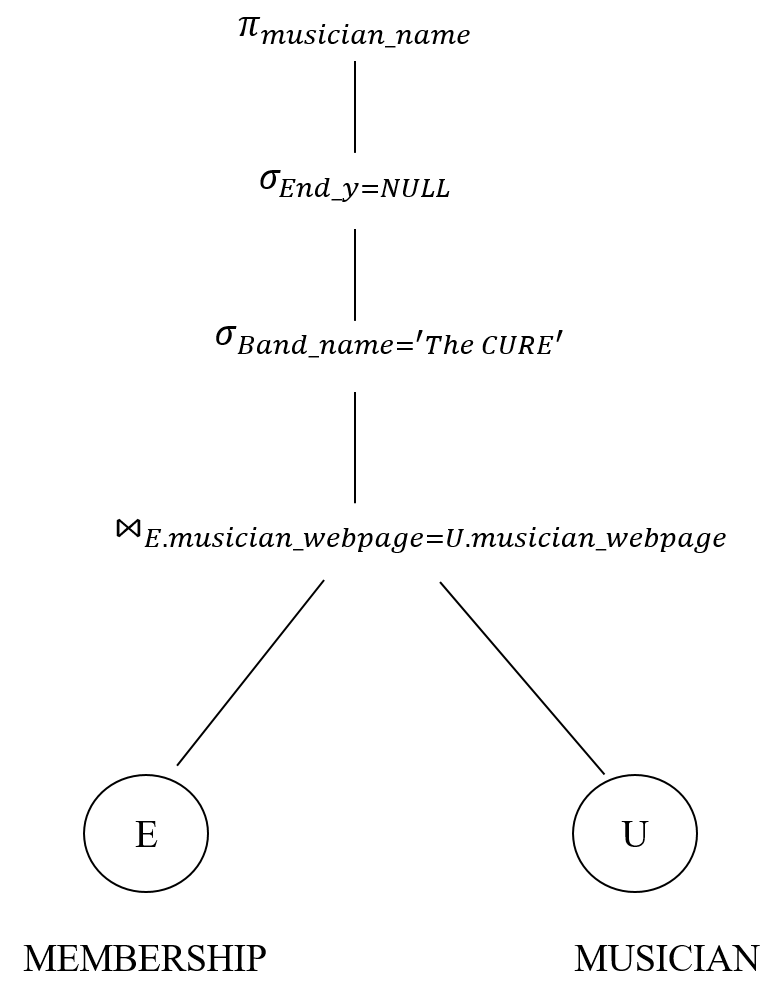


Figure 3: Relational Algebra Tree for SQL1

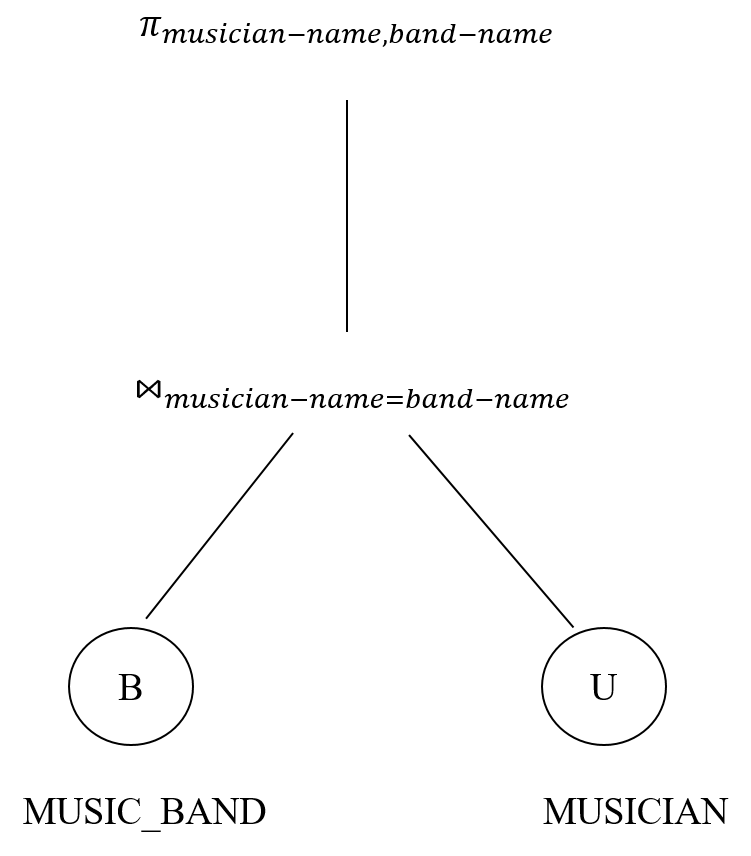


Figure 4: Relational Algebra Tree for SQL2

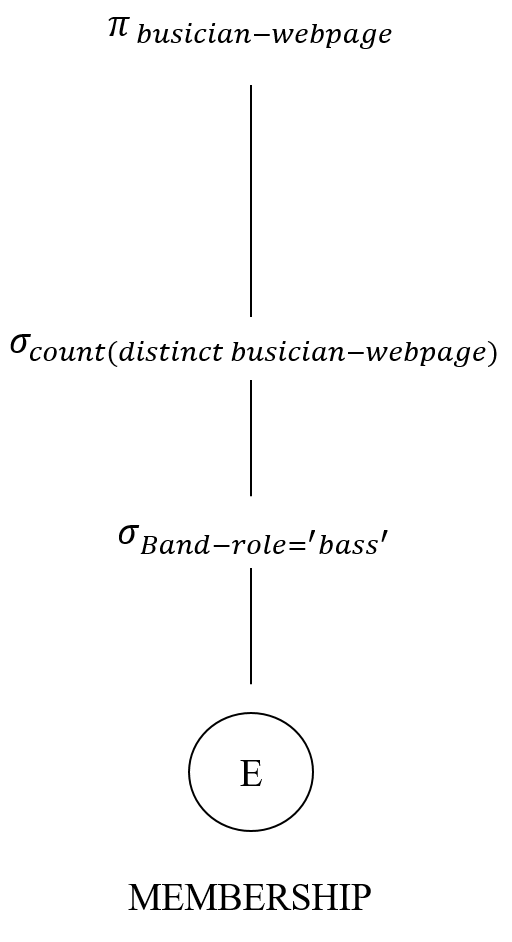


Figure 5: Relational Algebra Tree for SQL3

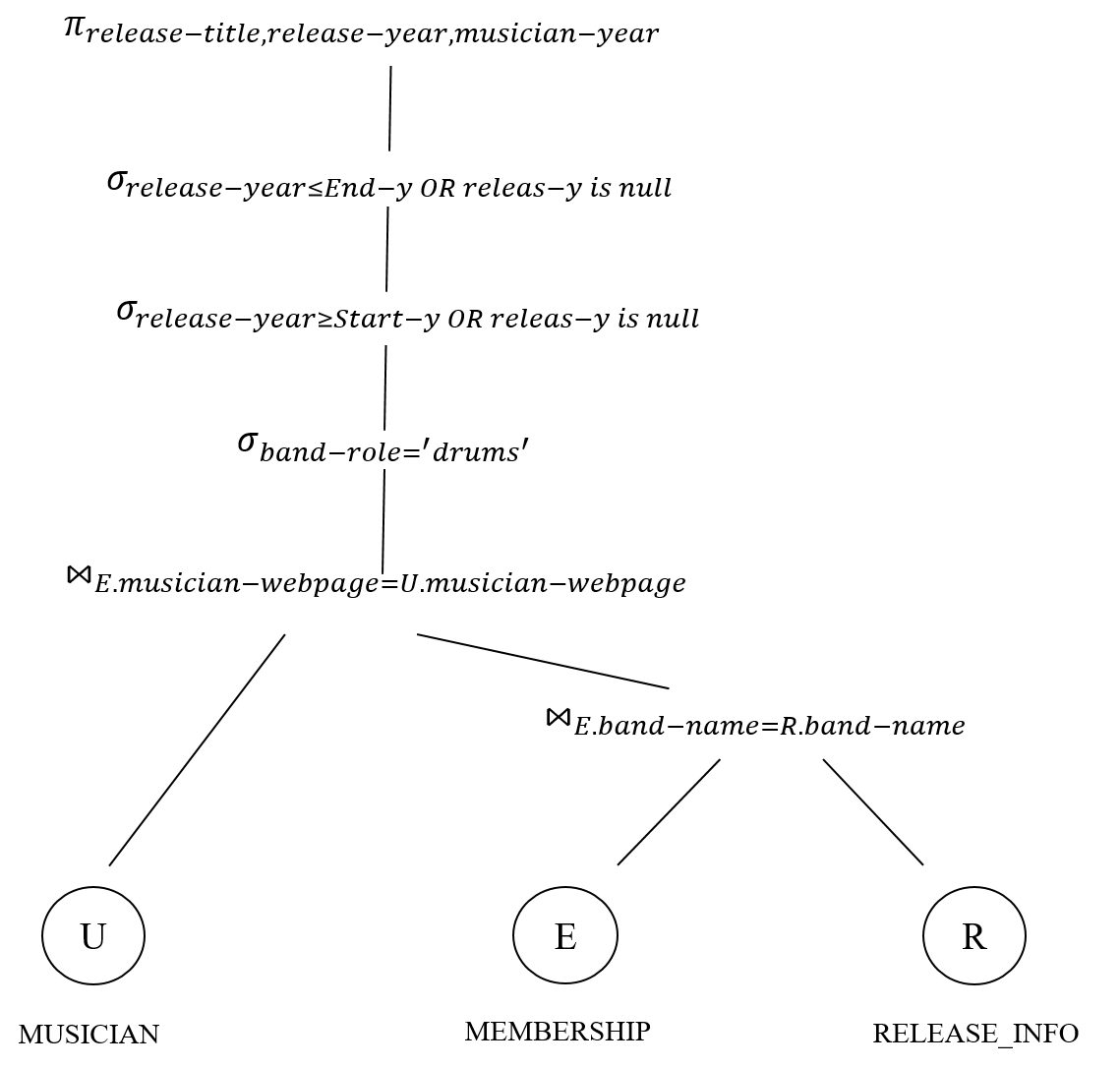


Figure 6: Relational Algebra Tree for SQL4

# Report 4.2: Heuristic Optimization

Rules applied on optimize SQL1:

Rule 2: Since the “Band\_name” is unique for each, it will be benefit to execute selection on band name first.

Rule 4:

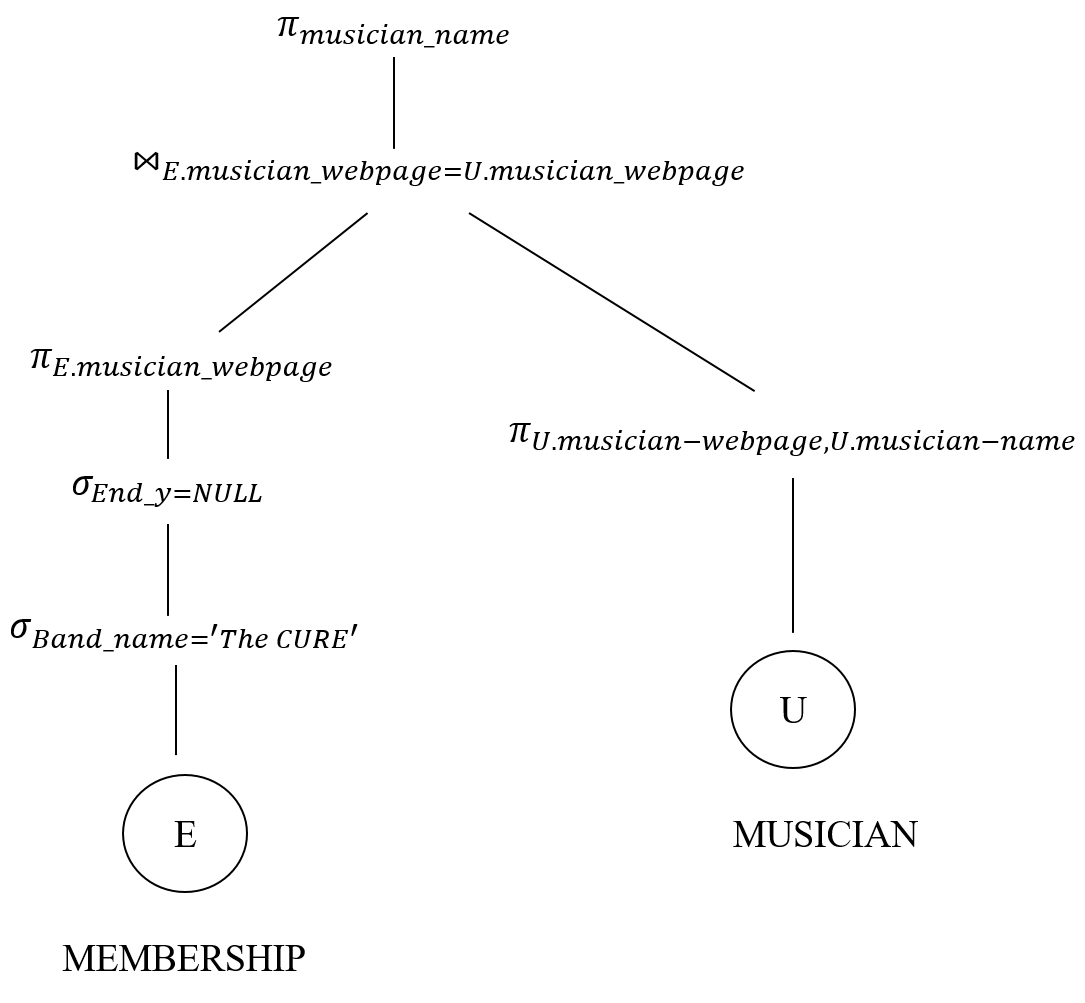


Figure 7: Optimal Relational Algebra Tree for SQL1

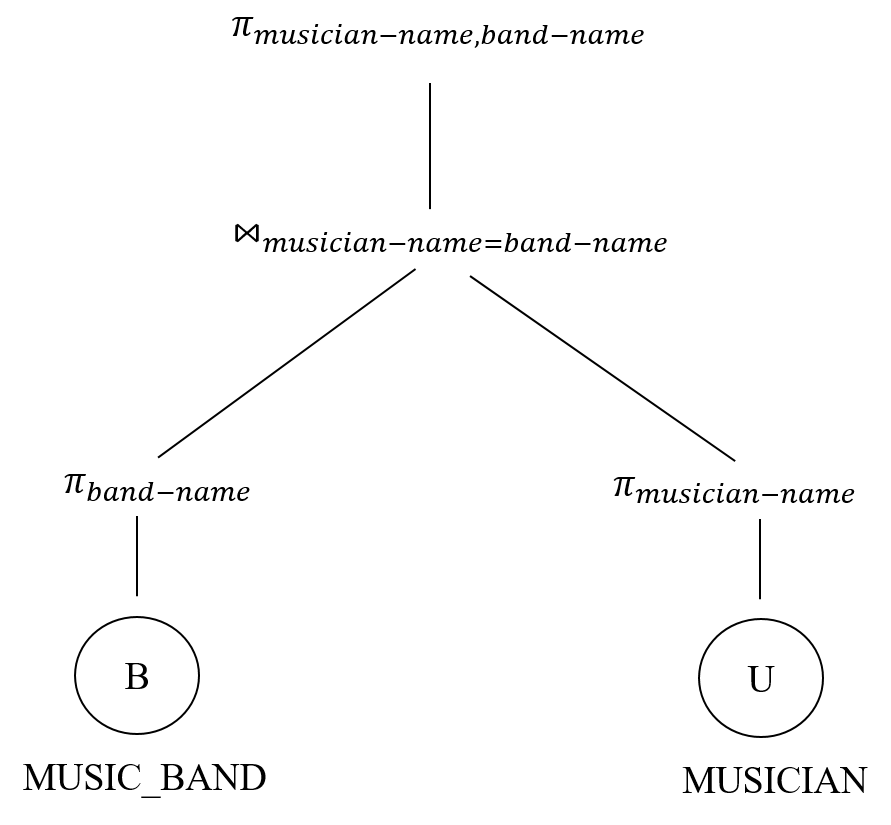


Figure 8: Optimal Relational Algebra Tree for SQL2

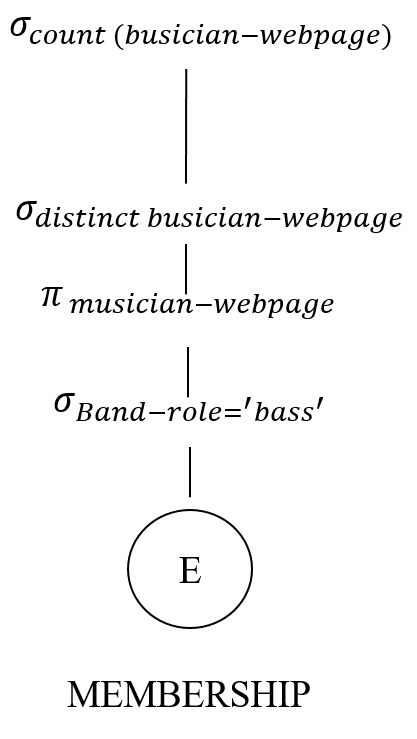


Figure 9: Optimal Relational Algebra Tree for SQL3

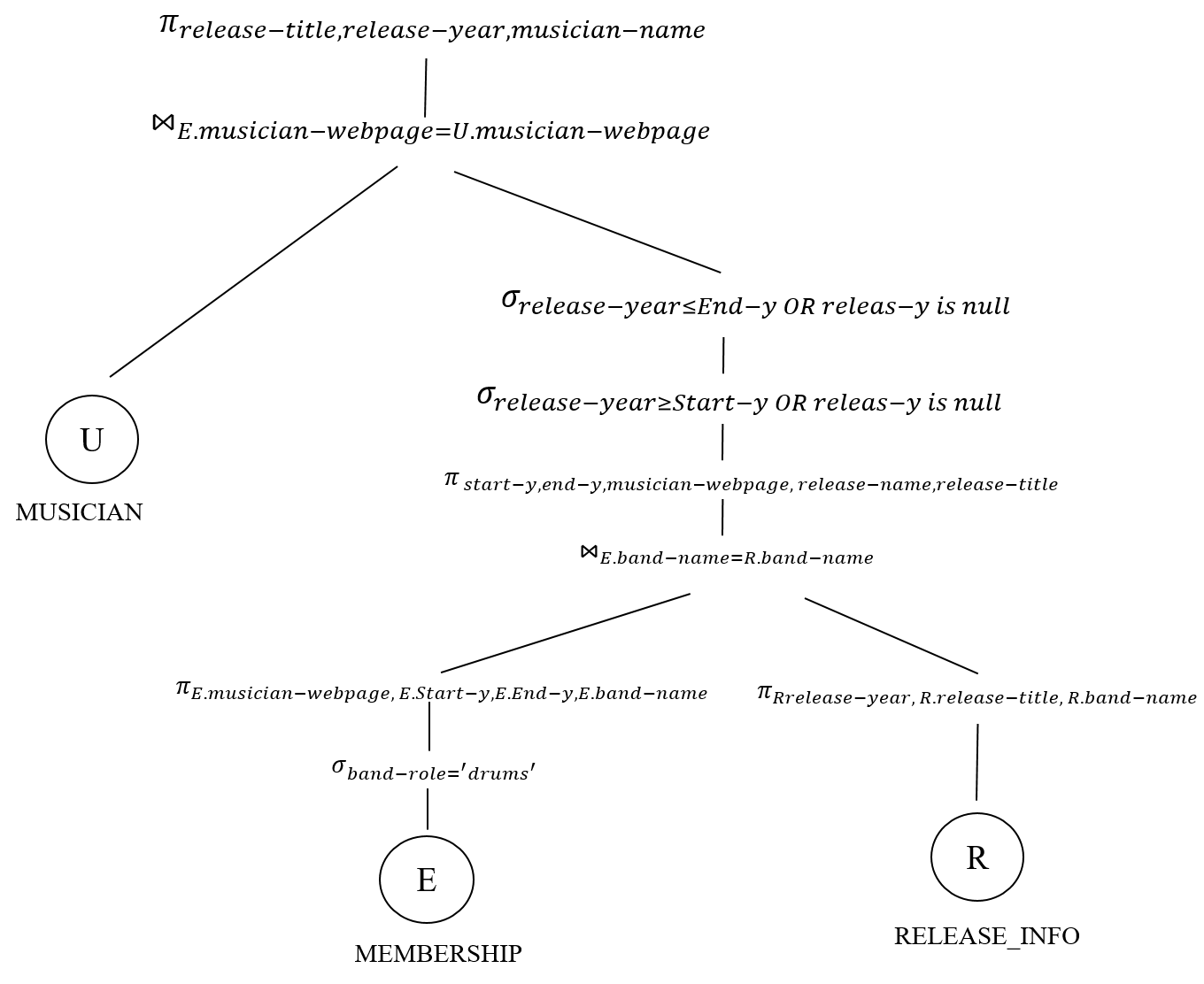


Figure 10: Optimal Relational Algebra Tree for SQL 4

# Report 5: JBDC Program

“DTA.java” only.

INSERT INTO MUSICIAN VALUES ('Ellison','www.elison.com');

INSERT INTO MUSIC\_BAND VALUES ('The CURE','www.thecure.com');

INSERT INTO MEMBERSHIP VALUES ('1111111111','www.elison.com','The CURE',null,null,'drums');

INSERT INTO MUSICIAN VALUES ('Kevin','www.kevin.com');

INSERT INTO MUSICIAN VALUES ('Edmund','www.edmund.com');

INSERT INTO MUSICIAN VALUES ('Elly','www.elly.com');

INSERT INTO MUSICIAN VALUES ('Bob','www.bob.com');

INSERT INTO MEMBERSHIP VALUES ('1111111112','www.kevin.com','The CURE',2017,2018,'vocals');

INSERT INTO MEMBERSHIP VALUES ('1111111113','www.edmund.com','The CURE',2017-10-10,null,'drums');

INSERT INTO MEMBERSHIP VALUES ('1111111114','www.elly.com','The CURE',2016-05-10,null,'bass');

INSERT INTO MEMBERSHIP VALUES ('1111111115','www.bob.com','The CURE',2016-01-01,2017-10-10,'vocals');

INSERT INTO MEMBERSHIP VALUES ('1111111116','www.bob.com','The CURE',2017-10-10,null,'vocals');

INSERT INTO MUSICIAN VALUES ('Andreas','www.andreas.com');

INSERT INTO MEMBERSHIP VALUES ('1111111118','www.andreas.com','Ellison',2015,2016,'vocals');

INSERT INTO MEMBERSHIP VALUES ('1111111119','www.andreas.com','Ellison',2017,2018,'drums');

# Bibliography

|  |  |
| --- | --- |
| [1] | T. Griffin, “Mark My Word: Trademark Your Band Name,” *Texas Review of Entertainment & Sports Law,* pp. 177-192, 2014. |