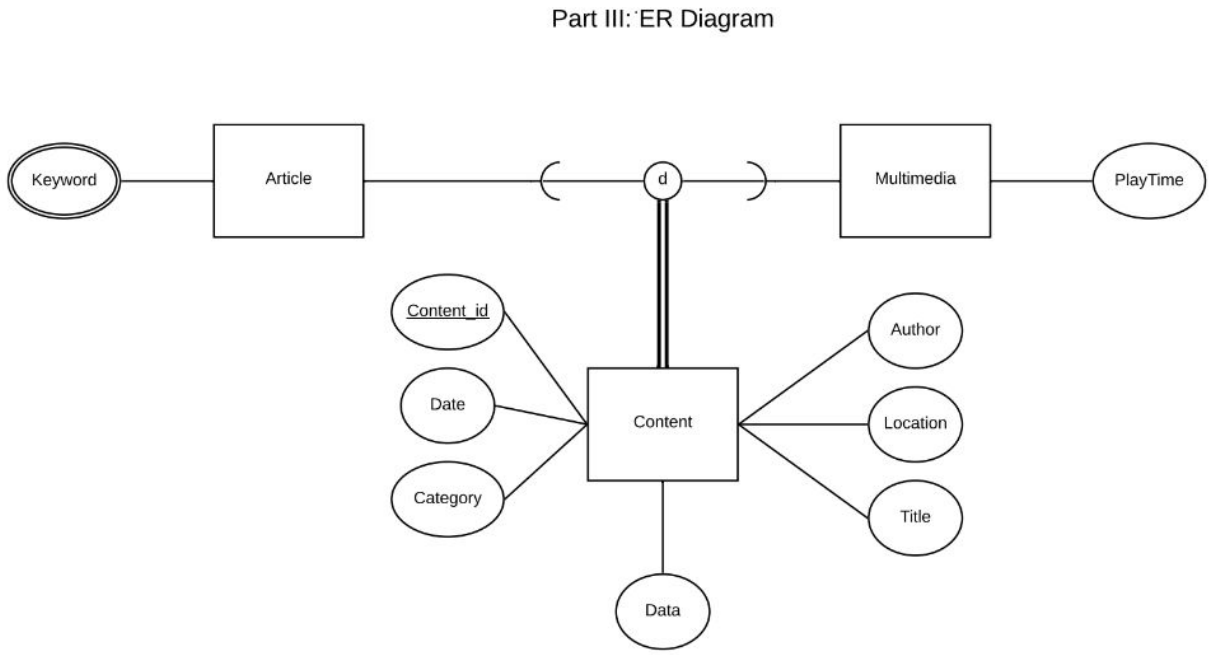
Team 10: Yash Dhayal, Zahir Johnson, Kevin Williams, Abhi Vempati, Mac Bivens-Tatum

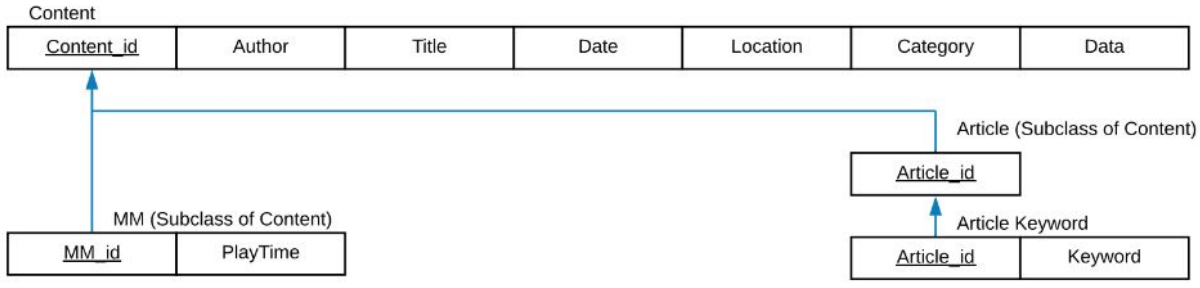
CSC 315-02

Stage III: Database Model

Entity-Relationship Diagram



Relational Schema



Without going into detail about database specifications for size, a rough estimate for records would be around 100 at most. Currently, there are 64 articles on the SR Hub website, starting from August 2019. However, there may be more information we may not have, such as podcasts and videos from the Civic Story site that may be worth putting into the SR Hub site. Furthermore, we can set a growth setting in case for the database to be able to handle more records for the space needed for the tables, views and other components.

Team 10: Yash Dhayal, Zahir Johnson, Kevin Williams, Abhi Vempati, Mac Bivens-Tatum

CSC 315-02

Stage 4: Design

Boyce-Codd Normal Form

The Content table wasn't in BCNF. While Content_id can determine all of the other attributes, Data can also determine both Category and Location. Thus, Content was split on Data.

The Article table consists of only article_id, so it cannot be normalized any further.

The Article_Keyword table consists of only article_id and keyword, so it cannot be normalized any further.

For the MM table, nothing needed to be changed since it was in BCNF.

Sheet of BCNF tables is at the end.

Views

Mapview contains Content_id and Location. This view would be used for the interactive NJ map.

```
CREATE VIEW Mapview AS
SELECT content_id, location
FROM CONTENT;
```

PieTags view contains Content_Id, Category, and Keywords. This view would be used for both the Pie chart and for listing out the tags.

```
CREATE VIEW PieTags AS
SELECT content_id, category, keywords
FROM CONTENT;
```

For PieTags view, we would need the following function to calculate the percentages per slice:

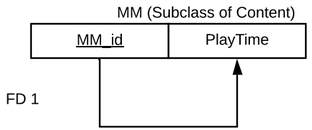
```
CREATE FUNCTION CategoryPercent(numerator PieTags.category%TYPE)
RETURNS NUMERIC(4, 2) AS $$
DECLARE
        need_val NUMERIC;
        total NUMERIC;
BEGIN
        SELECT COUNT(category) as total
        FROM PieTags;
        SELECT COUNT(numerator) as need_val
        FROM PieTags
        RETURN need_val / total;
END;
$$ LANGUAGE plpgsql;
```

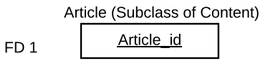
Tables, Functions & Transaction

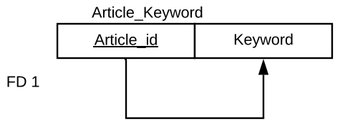
```
-- Creating Content table
CREATE TABLE CONTENT (
        Content_id      SERIAL          PRIMARY KEY,
        Author          VARCHAR(20)     NOT NULL,   -- Author last name
        Title           TEXT            NOT NULL,   -- Title of content
        Posting_date    DATE            NOT NULL,   -- Date content is posted
        Location        VARCHAR(20)     NOT NULL,   -- County relevant to content
        Category        VARCHAR(20)     NOT NULL,   -- Category of the content
        Data            TEXT            NOT NULL    -- Link to the content
);


-- Creating Multimedia table
CREATE TABLE MM(
        Playtime        TIME
) INHERITS (CONTENT);


-- Creating Article table. Will only go into this if inserted data is considered article
CREATE TABLE ARTICLE(           ) INHERITS (CONTENT);
```

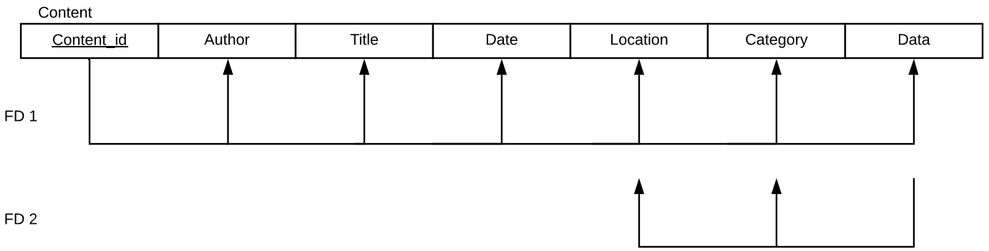
```
-- Creating article_keywords table
CREATE TABLE ARTICLE_KEYWORD(
        Keyword        TEXT                    NOT NULL;
) ;


-- function to insert into both master Content table and respective subclass table

CREATE FUNCTION MyInsert( _author VARCHAR(20), _title TEXT, _posting_date
DATE, _location VARCHAR(20), _category VARCHAR(20), _data TEXT, datatype
CHAR)
RETURNS void AS $$
BEGIN
        IF datatype = 'A' THEN                  -- enter into Article table
                INSERT INTO CONTENT(author, title, posting_date, location, category,
                data)
                VALUES(_author, _title, _posting_date, _location, _category, _data TEXT);

                INSERT INTO ARTICLE(author, title, posting_date, location, category,
                data)
                VALUES(_author, _title, _posting_date, _location, _category, _data TEXT);
        END IF;

        ELSEIF datatype = 'M' THEN          -- enter into MM table
                INSERT INTO CONTENT(author, title, posting_date, location, category,
                data)
                VALUES(_author, _title, _posting_date, _location, _category, _data TEXT);

                INSERT INTO MM(author, title, posting_date, location, category, data)
                VALUES(_author, _title, _posting_date, _location, _category, _data TEXT);
        END IF;

        ELSE  -- if data type is not accepted
                RAISE EXCEPTION 'Not acceptable data type provided: %', datatype;
        END IF;
END $$;
```

```
-- transaction for inserting

BEGIN;
MyInsert(_author, _title, _posting_date, _location, _category, _data); -- subject to change
-- ROLLBACK for situation when function raises exception
COMMIT;
```


MM (Subclass of Content)
MM_id
PlayTime
FD 1


Article (Subclass of Content)
FD 1
Article_id

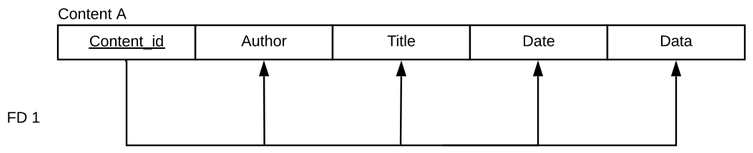

Article_Keyword
Article_id
Keyword
FD 1

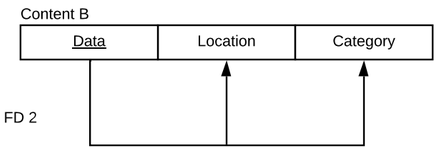

Content
Content_id
Author
Title
Date
Location
Category
Data
FD 1
FD 2

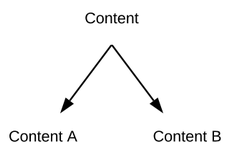

MM
MM


Article
Article


Article
Article


Content A
Content_id
Author
Title
Date
Data
FD 1


Content B
Data
Location
Category
FD 2


Content
Content A
Content B