Drew Neff & Ashton Reimer
Databases Project Proposal
1/31

We will build and populate a database of all bathrooms on Columbia's campus. We plan on storing meaningful data such as building, floor, handicap accessibility, and gender inclusivity as well as more superfluous data such the number of stalls, sinks, and visits from active database users. Bathrooms would be located by building and will be easily searchable so that users could find bathrooms when in need in addition to seeking out obscure bathrooms to review or verify for our database. As illustrated in our ER diagram, we anticipate three strong entities (Building, Bathroom, and User), one weak entity (Review), and two additional overlapping specializations of Bathroom (Residential and Unconfirmed). There are also nine relationships that we will integrate into our design.

Initially we plan to populate the database by physically walking through a few buildings on campus and making note of each bathroom and all of its attributes. Once we can confirm our database functions with a small amount of data we will import bathroom data we obtain from the CU Service Request Homepage. We know this method will not always provide us with data for all of the attributes so we anticipate allowing some attributes to be NULL or will populate the database with fake data if necessary. In the long run, users would be able to submit their own new bathrooms which would be stored as an Unconfirmed specialization of bathroom until a threshold of confirmations is reached through additional user verification.

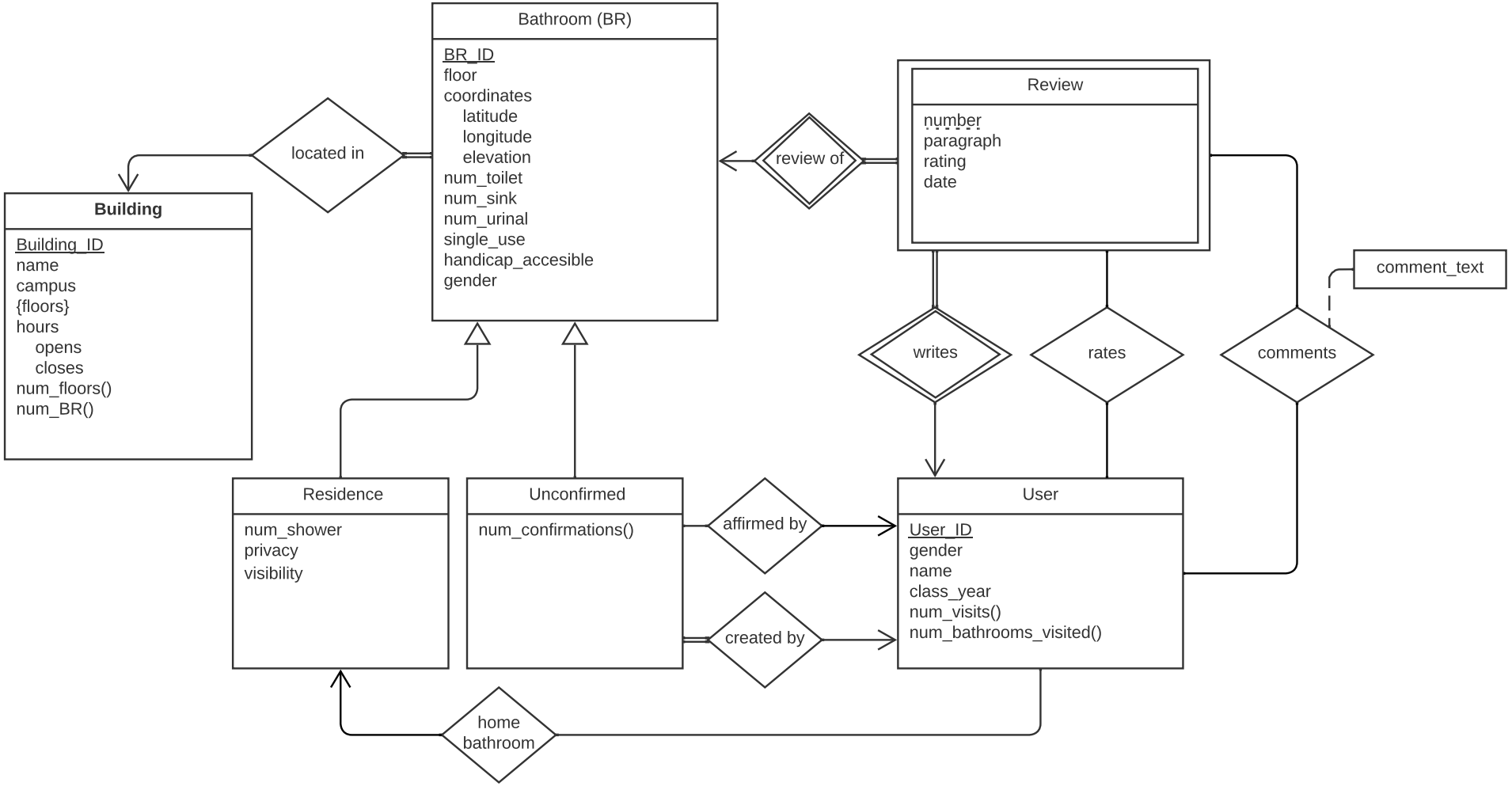
We plan on implementing the Web Front-End Option for part three which will allow users to interact with the database on a UI that lets them submit reviews for bathrooms they have visited as well as interact with other user reviews by rating them or setting their own home bathrooms for others to rate. Our interface will specify a difference in access for regular users and administrators. Users will only be able to create and rate reviews, suggest new bathrooms, and set their home bathroom. Administrators will have the ability to delete reviews as well as read and write access to all data in the database in addition to basic functionalities. Our interface will have a leaderboard which automatically queries and displays both users and bathrooms ranked by some of their attributes or other qualities we have yet to determine. We also plan to implement a button that returns to the user a randomized bathroom that the user has never visited within constraints of whether the bathroom is accessible for the user.

Our contingency plan is if a student drops out the scope of the project can be reduced by reducing both the total amount of data in the database to collect as well as by reducing the size of the model schema by removing or reducing a number of relationships that users may have with other entities. This will primarily be completed through the removal of the Unconfirmed bathroom type and the removal of the residence bathroom subtype. Thti will not only remove two entity types but also their additional relationships. With those changes the entity and relationship number will fall into the one person scope. Along with the removal of those entities a handful of changes to the overall functionality will occur. Home bathrooms will no longer exist

along with many of the more complicated database permissions type constraints we are expecting to see in later stages. In the barebones model it will simply be a database of bathrooms on campus that can be reviewed rather than additional bathrooms being able to be added. We would be disappointed in this loss of functionality so it is simply our contingency.

CUBR

Drew Neff & Ashton Reimer | February 10, 2023

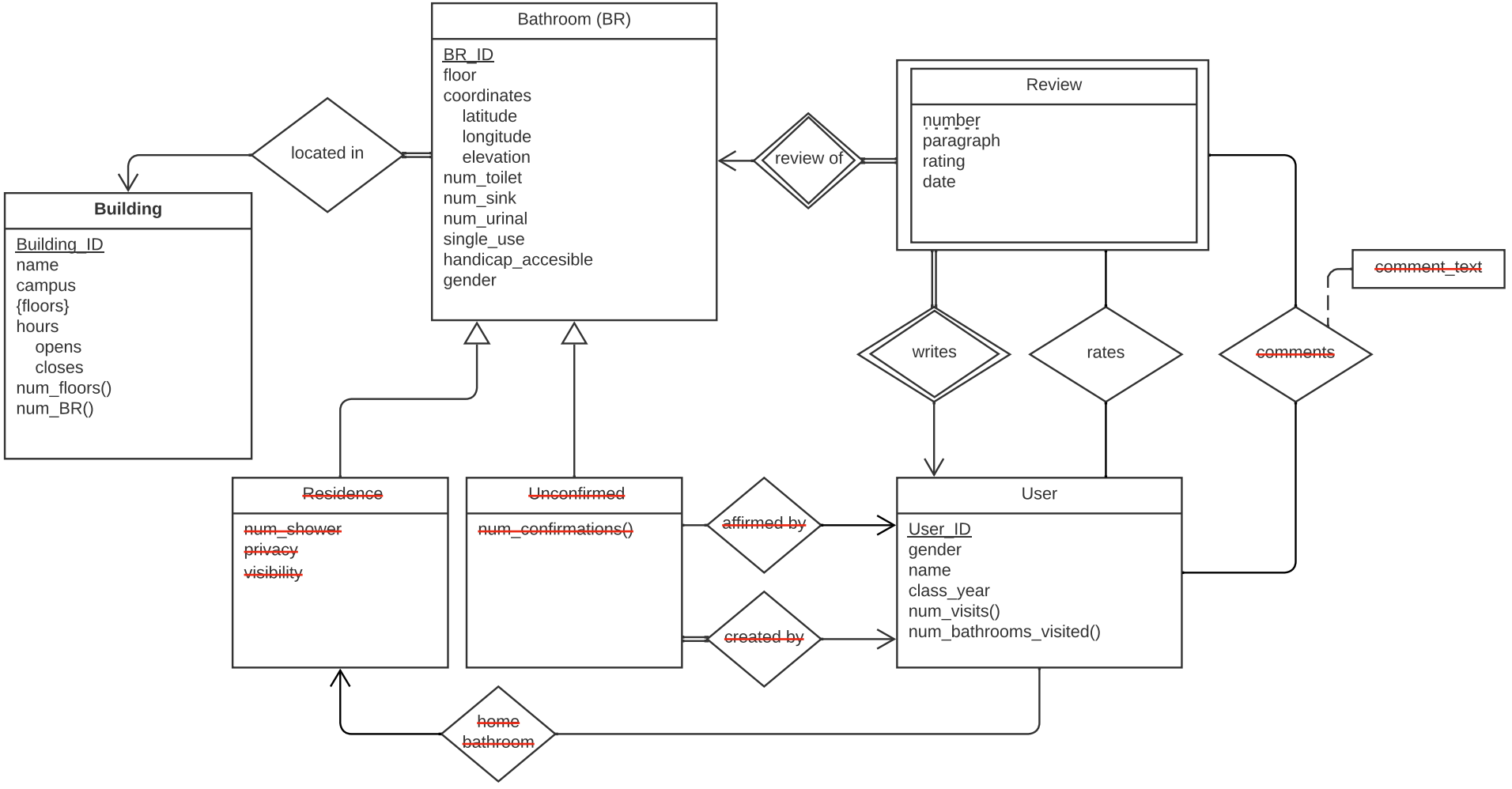


Notes:
We will implement a constraint in which users can only comment on reviews of their assigned “home bathroom”
Users cannot rate their own reviews

CONTINGENCY

CUBR

Drew Neff & Ashton Reimer | February 10, 2023



SQL Schema

We attempted to have as comprehensive of an SQL framework as possible, In trying to meet that goal while under the constraints of our current class knowledge we may have made a handful of small mistakes. We elected to make certain many-to-many relationships into tables. We did not know how to put in function variables so left those out. We also did not know how to do sub variables so we only used the children not the parents. Many permission constraints are not shown as we are yet to learn them.

```
Unset
CREATE TABLE bathroom (
BR_ID VARCHAR(8),
floor VARCHAR(20) NOT NULL,
Building_ID VARCHAR(8) NOT NULL,
lat_coord DOUBLE(10,7),
long_coord DOUBLE(10,7),
elevation_coord DOUBLE(10,7),
num_toilet INT,
num_sink INT,
num_urinal INT,
single_use BOOL,
handicap_accessible BOOL,
gender VARCHAR(8),
PRIMARY KEY (BR_ID),
FOREIGN KEY (Building_ID) REFERENCES building
);

CREATE TABLE building (
Building_ID VARCHAR(8),
name VARCHAR(20) NOT NULL,
campus VARCHAR(16) NOT NULL,
floors VARCHAR(MAX) NOT NULL,
hour_open TIME NOT NULL
hour_closed TIME NOT NULL,
PRIMARY KEY (Building_ID)
```

```
FOREIGN KEY (floors) REFERENCES floor_table);

CREATE TABLE review (
number INT NOT NULL,
BR_ID VARCHAR(8) NOT NULL,
User_ID VARCHAR(8) NOT NULL,
paragraph TINYTEXT,
rating BOOL,
date DATETIME NOT NULL,
PRIMARY KEY (BR_ID, User_ID, number),
FOREIGN KEY (User_ID) REFERENCES user)
ON DELETE CASCADE,
FOREIGN KEY (BR_ID) REFERENCES bathroom
ON DELETE CASCADE
);


CREATE TABLE user (
User_ID VARCHAR(8) NOT NULL,
gender VARCHAR(8),
name VARCHAR(20) NOT NULL,
class_year NUMERIC(4,0),
PRIMARY KEY (User_ID)
);

CREATE TABLE residence (
num_shower INT,
privacy BOOL NOT NULL,
visibility BOOL NOT NULL)
INHERITS bathroom;


CREATE TABLE unconfirmed (
```

```
num_confirmations INT NOT NULL,
User_ID VARCHAR(8) NOT NULL,
FOREIGN KEY (User_ID) REFERENCES user)
INHERITS bathroom;


CREATE TABLE ratings (
User_ID_rater VARCHAR(8) NOT NULL,
User_ID_ratee VARCHAR(8) NOT NULL,
BR_ID VARCHAR(8) NOT NULL,
rating BOOL NOT NULL,
number INT NOT NULL,
PRIMARY KEY (BR_ID, User_ID_rater, number, User_ID_ratee,
FOREIGN KEY (BR_ID) REFERENCES bathroom,
FOREIGN KEY (number) REFERENCES review,
FOREIGN KEY (User_ID_rater) REFERENCES user,
FOREIGN KEY (User_ID_ratee) REFERENCES user);

CREATE TABLE comment (
User_ID_commenter VARCHAR(8) NOT NULL,
User_ID_reviewer VARCHAR(8) NOT NULL,
BR_ID VARCHAR(8) NOT NULL,
comment_txt TINYTEXT NOT NULL,
number INT NOT NULL,
PRIMARY KEY (BR_ID, User_ID_reviewer, number,
User_ID_commenter),
FOREIGN KEY (BR_ID) REFERENCES bathroom,
FOREIGN KEY (number) REFERENCES review,
FOREIGN KEY (User_ID_commenter) REFERENCES user,
FOREIGN KEY (User_ID_reviewer) REFERENCES user);

CREATE TABLE affirmations (
BR_ID VARCHAR(8),
```

```
USER_ID VARCHAR(8),
affirmation_status BOOL,
PRIMARY KEY (BR_ID, User_ID),
FOREIGN KEY (BR_ID) REFERENCES unconfirmed,
FOREIGN KEY (User_ID) REFERENCES user);

CREATE TABLE homebathrooms (
User_ID VARCHAR(8),
BR_ID VARCHAR(8),
PRIMARY KEY (User_ID),
FOREIGN KEY (BR_ID) REFERENCES residence,
FOREIGN KEY (USER_ID) REFERENCES user);
```