Kramer Johnson

Homework 3

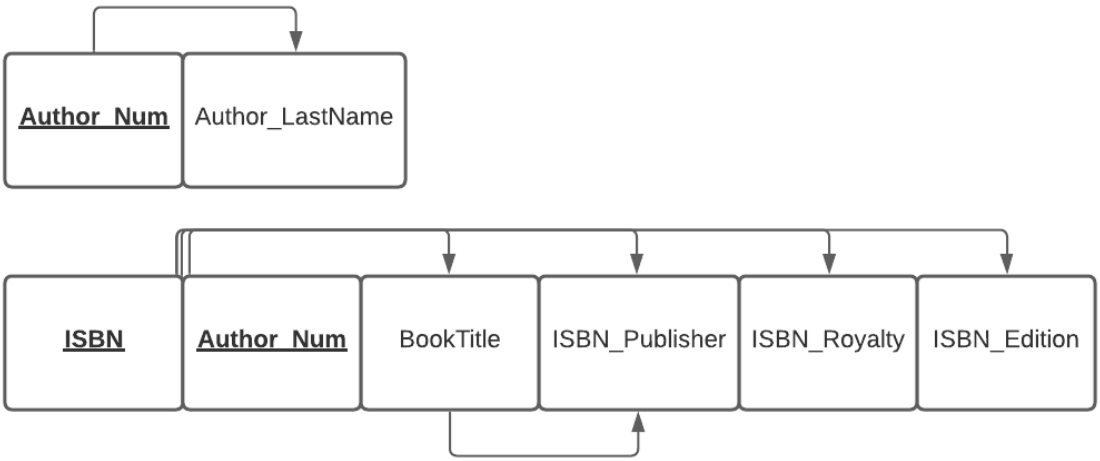
Database Systems 5021 02

Chapter 6 Review Questions:

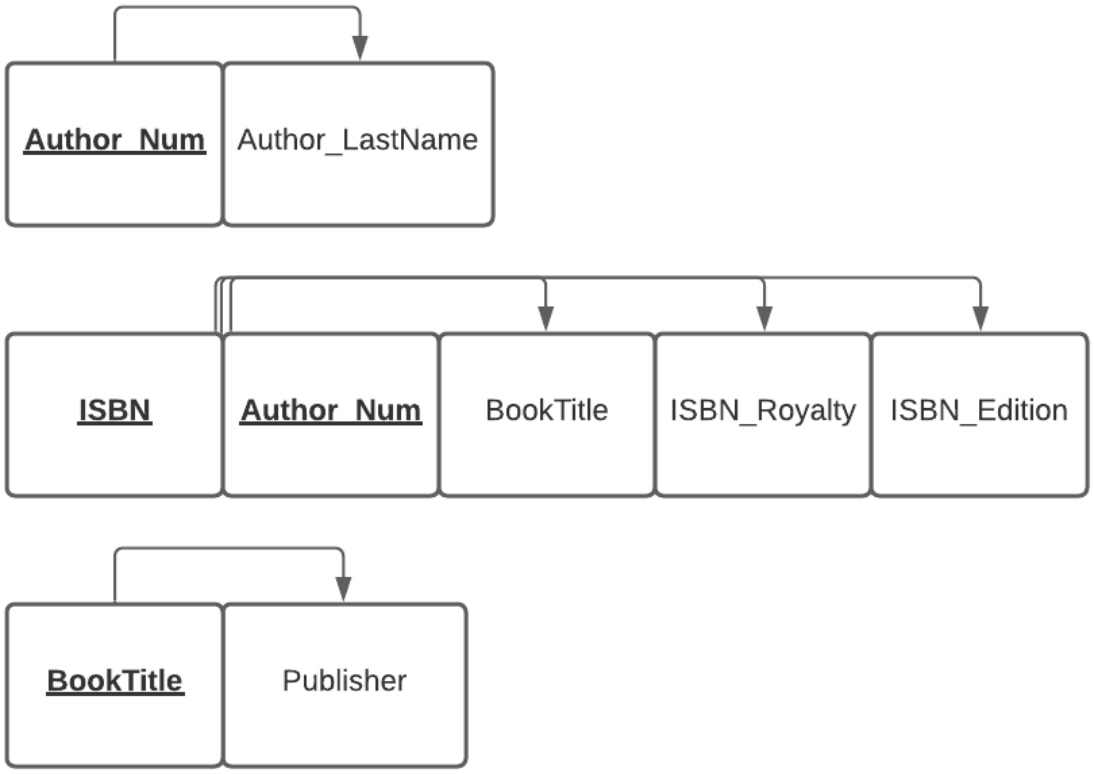
1. Normalization is the process used to reduce data redundancies and anomalies in data by separating larger tables into their separate entities. For example, if a single table contains data about customer's and agents, there can be redundancies in customer and agent names. These redundancies can be removed by separating the two entities into a customer table and an agent table.

4. A table is in 3NF when it is in 2NF and has no transitive dependencies. A transitive dependency is when an attribute in the table is dependent on another, non-prime, attribute. A job charge-rate attribute may be transitively dependent on a job-class attribute when the primary key does not include either attribute. A 3NF table will separate this job into its own entity to remove the dependency.

7. a.



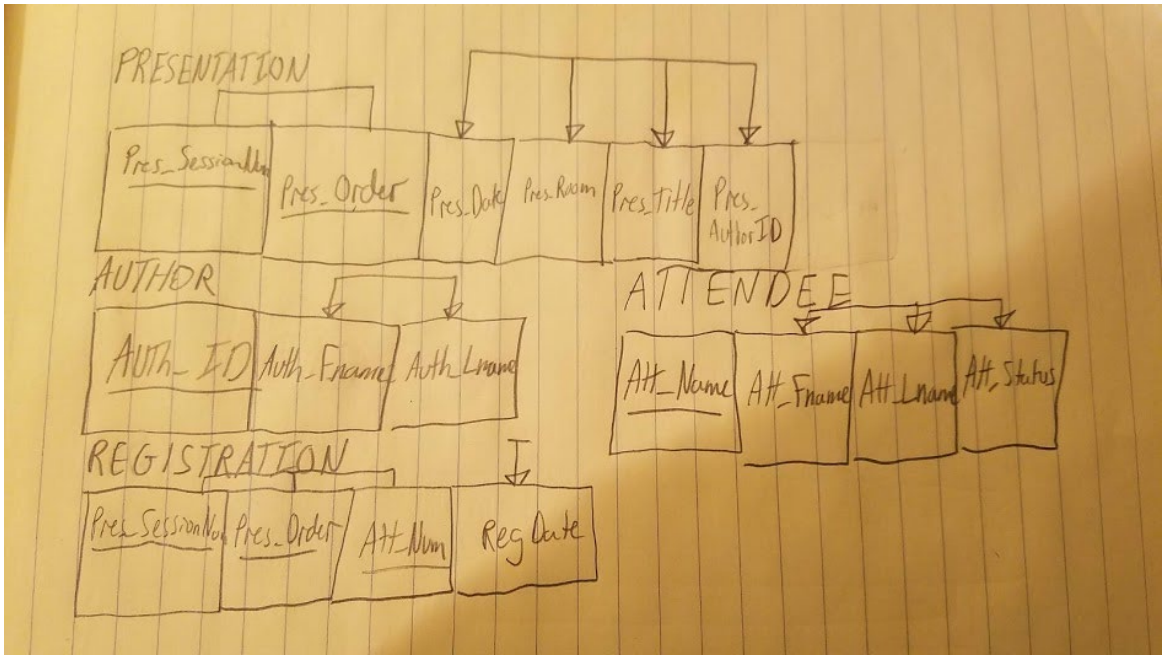
b.



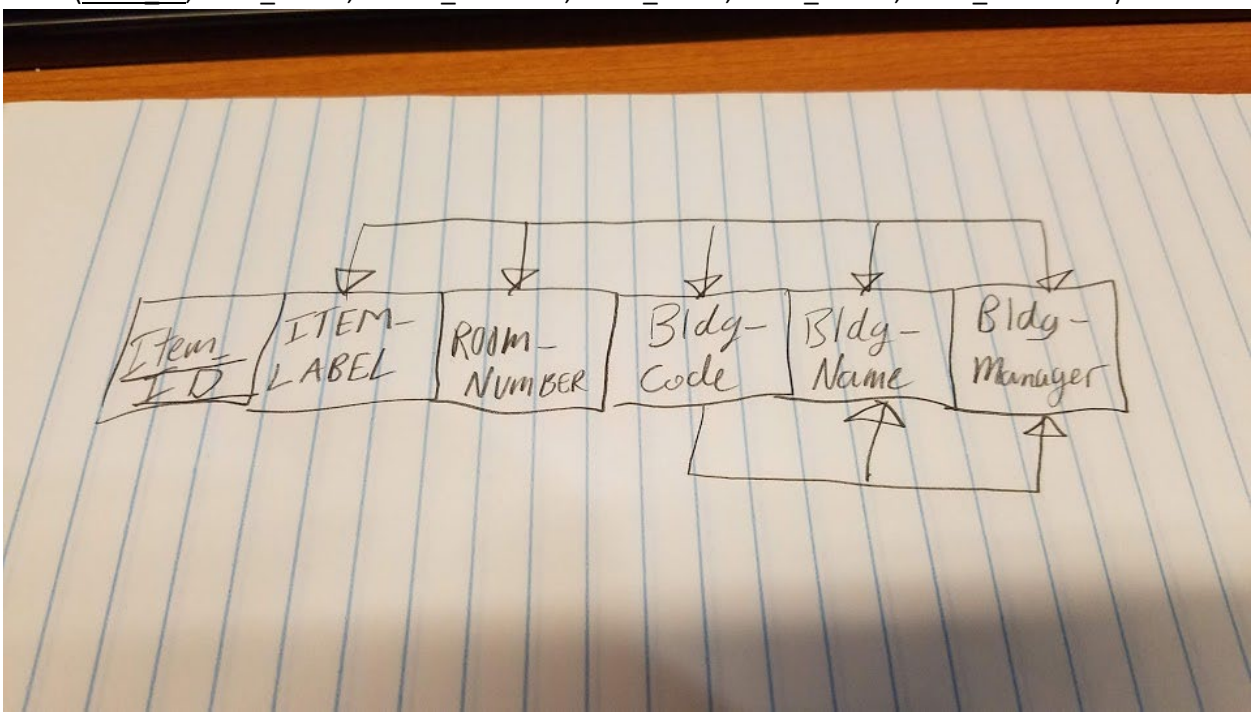
10. The three anomalies likely to result from data redundancy are insertion, update, and deletion anomalies. Insertion anomalies result from adding data to the database that has an absence of other required data. Such as attempting to add a customer record to a database, but that record requires an agent_num and the customer has not yet been assigned an agent. An update anomaly occurs when you need to update multiple records to change one. For example, if a customer's name occurs in multiple locations you would need to update multiple records to change their name. A deletion anomaly occurs when deletion of one set of data results in the unintended deletion of other data. Such as in the same customer/agent record, if that customer is an agent's only customer, and the customer is deleted, the agent would be unintentionally deleted as well.

13. A table that has only a single attribute primary key will automatically be in 2NF when it is in 1NF because a partial dependency is impossible when the primary key is made up of a single attribute. A partial dependency is when an attribute is dependent on only a subset of the primary key. A single attribute primary key cannot have a subset and therefore cannot have any of the partial dependencies that need to be fixed to become 2NF.

Problems:

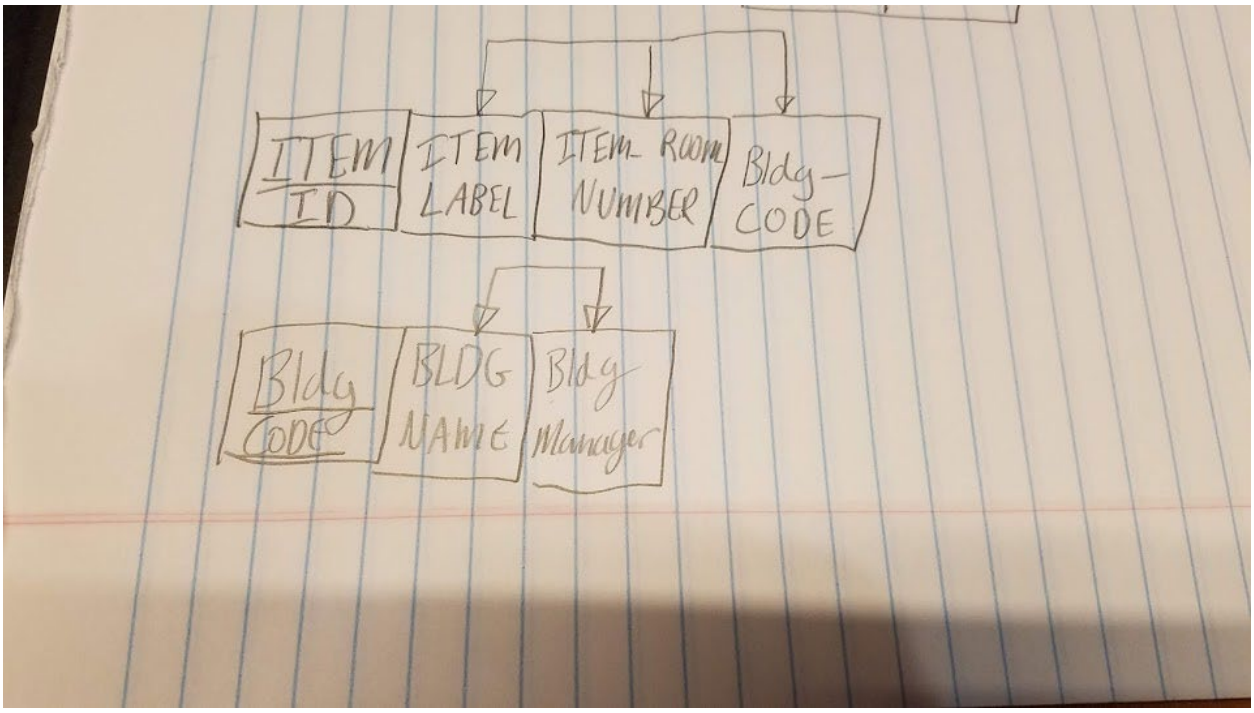
2. 

5. a. (**ITEM_ID**, ITEM_LABEL, ROOM_NUMBER, BLDG_CODE, BLDG_NAME, BLDG_MANAGER)

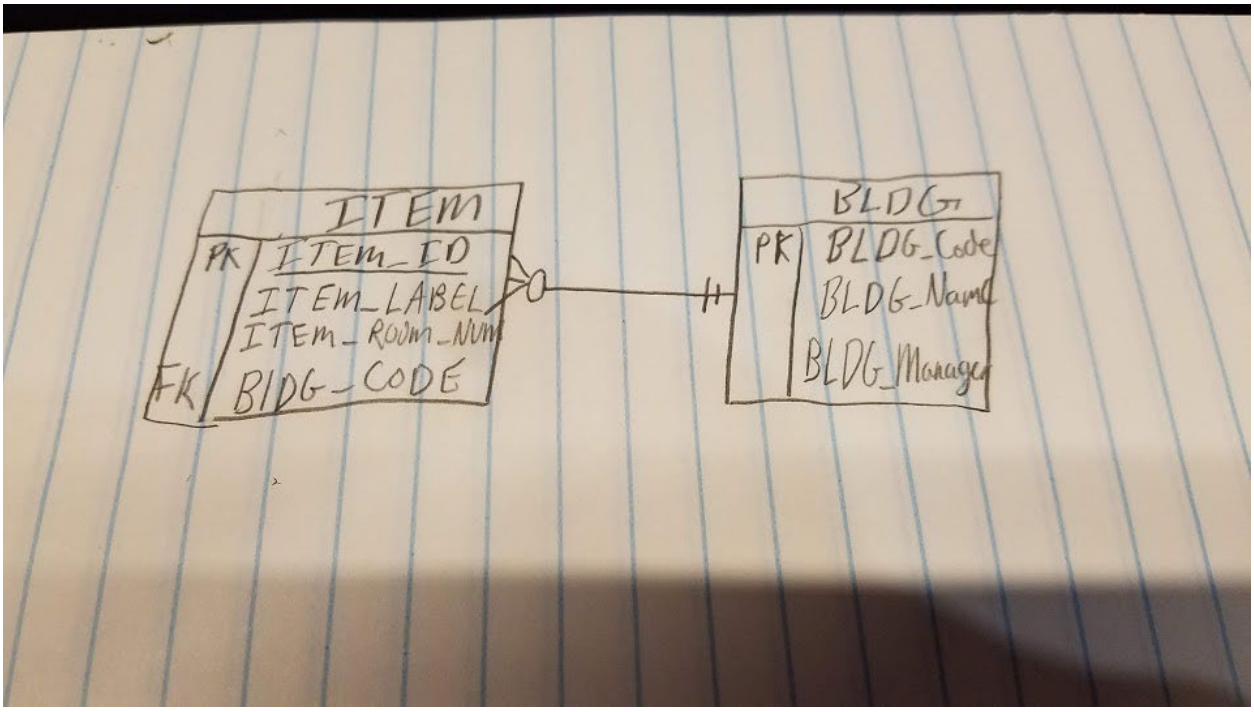


b. (**ITEM_ID**, ITEM_LABEL, ITEM_ROOM_NUMBER, BLDG_CODE)

(**BLDG_CODE**, BLDG_NAME, BLDG_MANAGER)


ITEM ID
ITEM LABEL
ITEM_ROOM NUMBER
Bldg-CODE
Bldg CODE
BLDG NAME
Bldg Manager

c.


ITEM
PK
ITEM_ID
ITEM_LABEL
ITEM_ROOM_NUM
FK
BLDG_CODE
BLDG
PK
BLDG_Code
BLDG_Name
BLDG_Manager

Chapter 7 Review Questions:

1. A SQL WHERE clause is evaluated prior to any grouping done in the statement and applies to expressions of individual rows. On the other hand, a HAVING clause is applied to the output of grouping. Specifically, a WHERE clause can filter by any column and values in the table while a HAVING clause can only filter by columns that are included in the group. For example,

```
SELECT      Invoice_ID, Invoice_Price
FROM        Sales.Invoices
WHERE       Invoice_date > CURRDATE()
GROUP BY Invoice_ID
HAVING      Invoice_Price > 200
```

Is a valid statement because the WHERE clause does not need to deal with only values included in the group while the HAVING clause must deal with either invoice_id or invoice_price.

Also, a WHERE clause cannot contain an aggregate function but the HAVING clause can.

4. If you do not need to perform mathematical operations on a numerical attribute (such as an invoice number, zip code, or phone number) then a character data type should be used to prevent someone from doing so and getting inappropriate data results.

7. WHERE V_STATE = 'TN' OR V_STATE = 'FL' OR V_STATE = 'GA';

10. The COUNT aggregate function returns the number of attributes that are found in a column. It reduces their values into a single row that displays the number of non-null values.

```
SELECT   COUNT(INV_NUM)
FROM    INVOICE
```

Could result in a single row that lists 25 as the total number of non-null occurrences of INV_NUM.

The SUM aggregate function adds the values found in an attribute and also displays them in a single row.

```
SELECT   SUM(INV_PRICE)
FROM    INVOICE
```

Could result in a single row displaying 24,355 as the total value off all the invoice prices added together.

Problems:

1.

```
CREATE TABLE EMP_1 (
   EMP_NUM          CHAR(3)          PRIMARY KEY,
   EMP_LNAME        VARCHAR(15)   NOT NULL,
   EMP_FNAME        VARCHAR(15)   NOT NULL,
   EMP_INITIAL      CHAR(1),
   EMP_HIREDATE   DATE,
   JOB_CODE         CHAR(3),
   FOREIGN KEY  (JOB_CODE) REFERENCES JOB ON UPDATE CASCADE
);
```

4. COMMIT;

7. ROLLBACK;