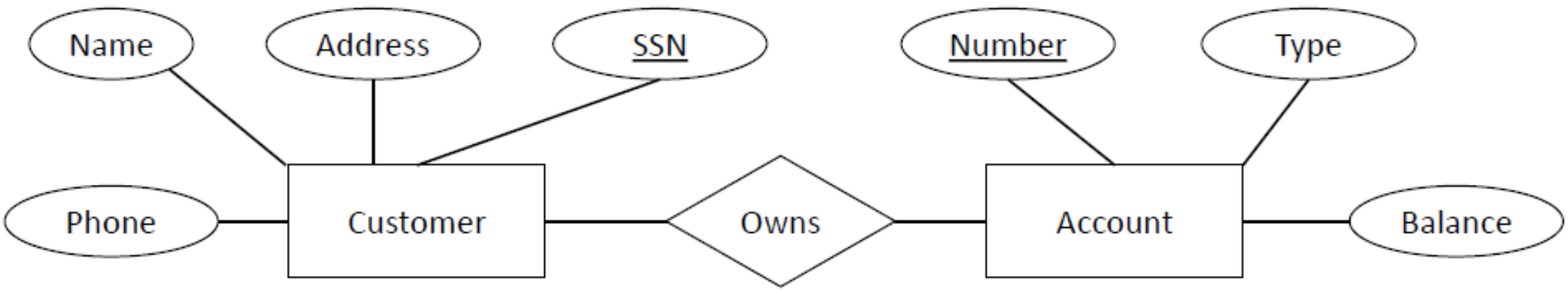
Conceptual & Physical Data Modeling

Database Design for a Bank

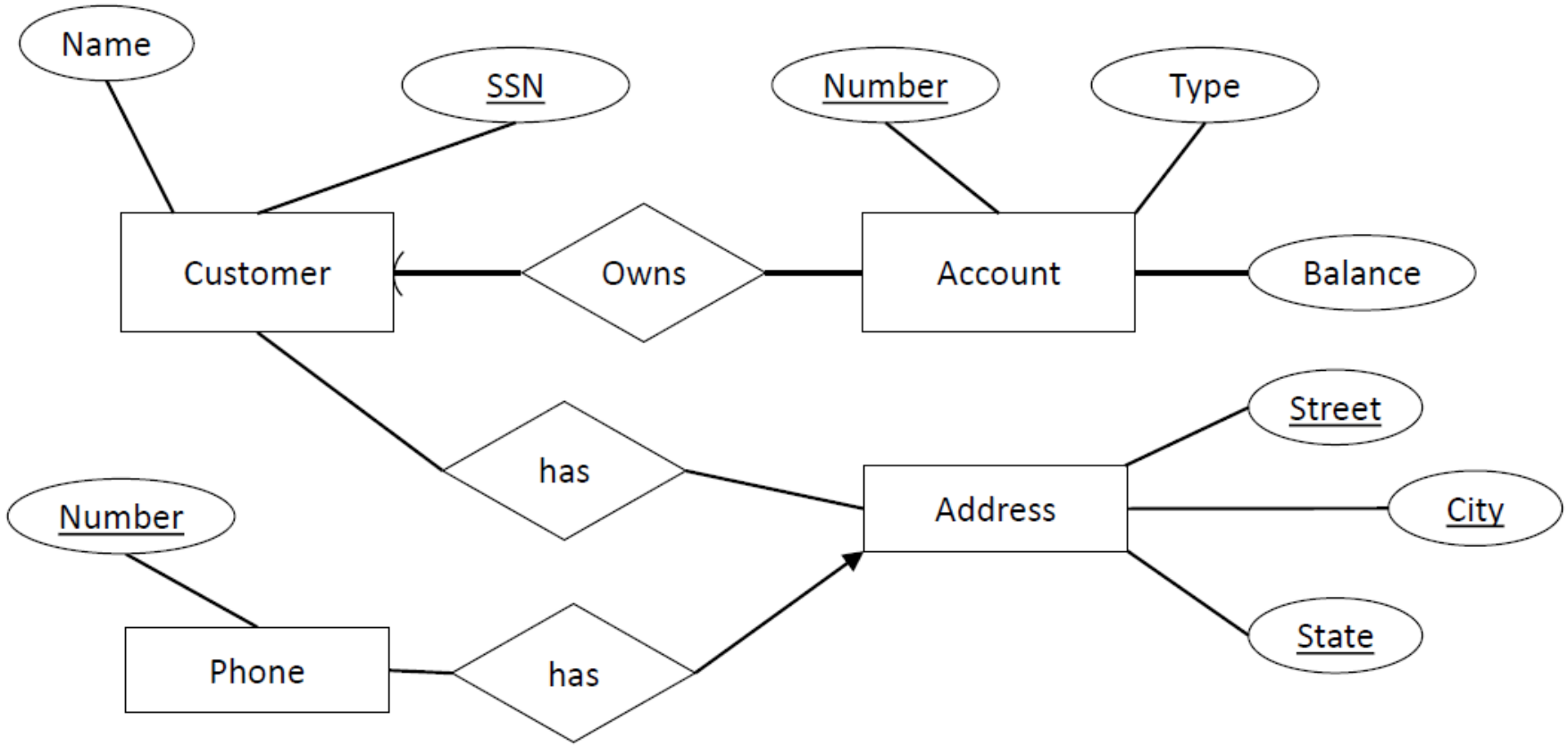
Draw an E/R diagram for the database incorporating the following requirements. Information about a customer includes their name, address, phone, and Social Security number. A customer is uniquely identified by his SSN. An account has a number, a type (e.g., saving, current) and a balance. The account number is used to uniquely identify an account. We also need to record the customer(s) who own an account. A customer may own more than one account. Be sure to include arrows where appropriate, to indicate the multiplicity of a relationship. Also, do not forget to specify the keys.


Name
Address
SSN
Number
Type
Phone
Customer
Owns
Account
Balance

Conceptual & Physical Data Modeling

Database Design for a Bank (Part II)

Redraw the ER diagram, from your solution to the previous question, incorporating the following additional requirements. An account has exactly one customer. Allow a customer to have a set of addresses, which are street-city-state triples, and at each address there is a set of phones. Note that a phone number can have at most one address associated with it. Be sure to include arrows where appropriate, to indicate the multiplicity of a relationship. Also, do not forget to specify the keys.


Name
SSN
Number
Type
Customer
Owns
Account
Balance
has
Street
Address
City
Number
State
Phone
has

3 Functional Dependency - 10 points

1. Consider the relation $R(A, B, C, D, E)$ with FDs $\mathcal{F} = \{AB \to C, C \to BD, E \to A, A \to D\}$.

   (a) [2.5] Find $\{B, E\}^+$.

   (b) [2.5] Show that $CE \to AD \in \mathcal{F}^+$ using Armstrong's axioms.

2. [5] Find a BCNF decomposition of R. Is it unique?

3.1

(a)

$\because E \to A \ , \ A \to D \ \therefore BE \to ABDE$

$\because AB \to C \ \therefore \{B, E\}^+ = \{A, B, C, D, E\}$

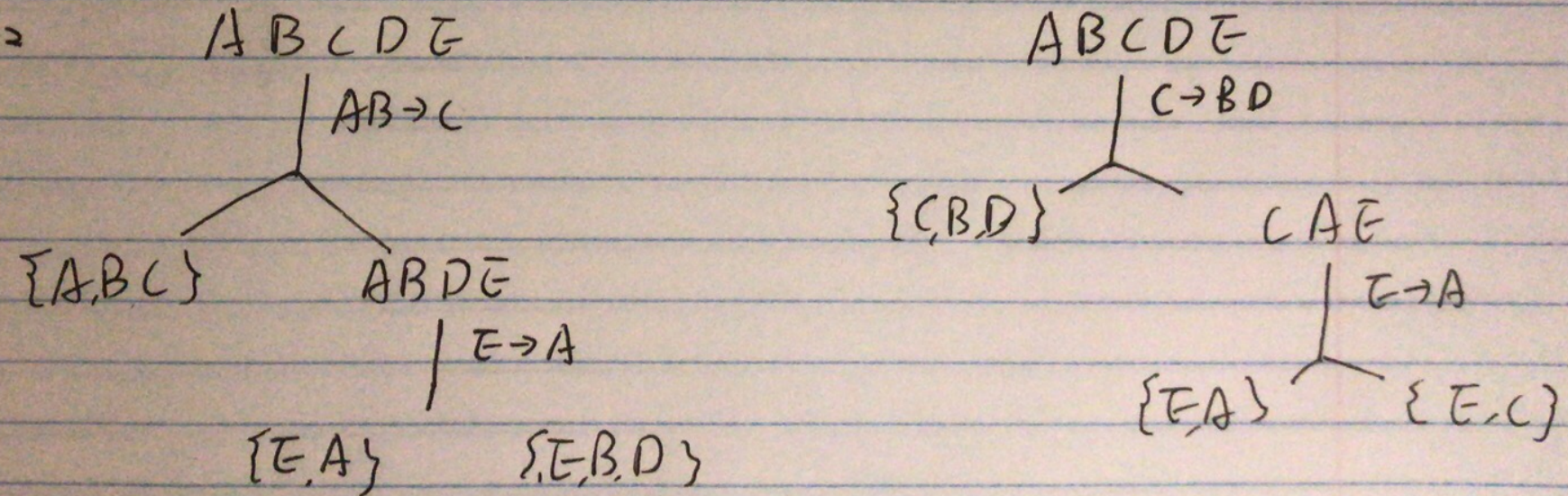
(b) $C \to BD \Rightarrow CE \to BDE$ (aug)

$BDE \to E$ (ref)

$BDE \to E, E \to A \Rightarrow BDE \to AD$ (trans, aug)

$CE \to BDE, BDE \to A \Rightarrow CE \to AD.$

3.2



Querying in Relational & Non-Relational Databases

SQL

Twitter is an online social networking and microblogging service that enables users to send and read “tweets”, which are text messages limited to 140 characters. Users may subscribe to other users' tweets - this is known as following and subscribers are known as followers.

Let's consider a small version of the Twitter database. We will use three relations:

- Users(UserID, Name, EMail), which stores the information about individual users.
- Tweets(TweetID, UserID, Tweet), which stores the users tweets.
- Followers(UserID, FollowUserID), which represents the following relationship.

User with UserID is followed by user with FollowUserID.

Users

UserID	Name	EMail
neo	Thomas Anderson	one@matrix.net
morph	Morpheus	morph@nebuchadnezzar.net
tri	Trinity	one@matrix.net
smith	Agent Smith	smith@matrixagents.net

Tweets

TweetID	UserID	Tweet
101	one	Going to meet Oracle !!
102	tri	Best of Luck !!
103	morph	In the Matrix
104	smith	Will catch all of them

Followers

UserID	FollowUserID
neo	morph
neo	tri
neo	smith
morph	smith
tri	smith
morph	neo

Conceptual & Physical Data Modeling

SQL

1. List tweets of all the users that are followed by “smith”.

2. List the userids that are not followed by any one.

Conceptual & Physical Data Modeling

SQL

3. We currently do not have a check for distinct email address. Hence it is possible to have multiple users with same email address. Write a query to get a list UserIDs of such users. i.e., a list of UserIDs where the email address is common to atleast one other user.

Conceptual & Physical Data Modeling

SQL

1. List tweets of all the users that are followed by “smith”.

$$\pi_{Tweet}\left(\sigma_{FollowUserID='smith'}(Followers \bowtie Tweets)\right)$$

OR

$$\pi_{Tweet}\left(\sigma_{FollowUserID='smith'}(Followers) \bowtie Tweets)\right)$$

2. List the userids that are not followed by any one.

$$\pi_{UserID}(Users) - \pi_{UserID}(Followers)$$

Conceptual & Physical Data Modeling

SQL

3. We currently do not have a check for distinct email address. Hence it is possible to have multiple users with same email address. Write a query to get a list UserIDs of such users. i.e., a list of UserIDs where the email address is common to atleast one other user.

```
SELECT DISTINCT UserID
FROM   Users u1, Users u2
WHERE  u1.email = u2.email
AND    u1.userid <> u2.userid
```

```
SELECT UserID
FROM   Users u1
WHERE  Email in  (
SELECT   Email
FROM     Users
GROUP BY EMail
HAVING   COUNT(1) > 1)
```

```
SELECT UserID
FROM   Users u1
WHERE  Email in  (
SELECT   Email
FROM     Users
GROUP BY EMail
HAVING   COUNT(1) > 1)
```

```
SELECT   UserID FROM     Users
GROUP BY EMail HAVING   COUNT(1) > 1
```

(????)

Problem 3 (*16 points*) Indexing

(1) Consider the following B+Tree of order 4 (i.e., n=4, each index can hold n keys and n + 1 pointers), shown in the figure below:

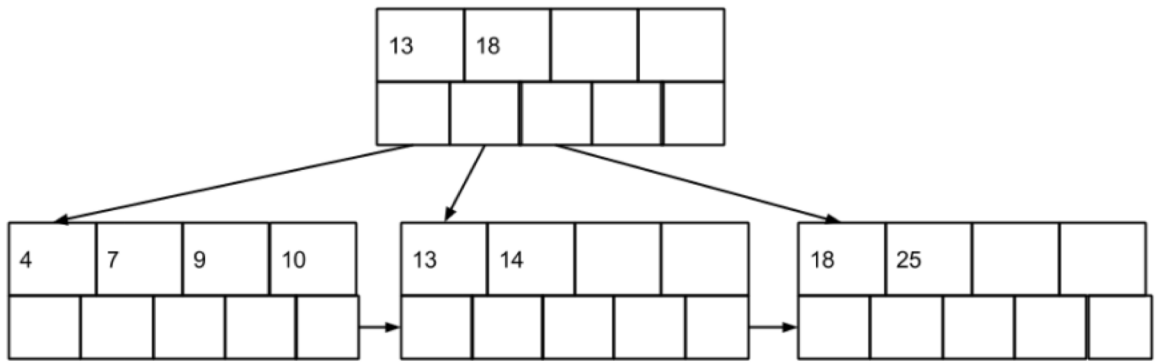
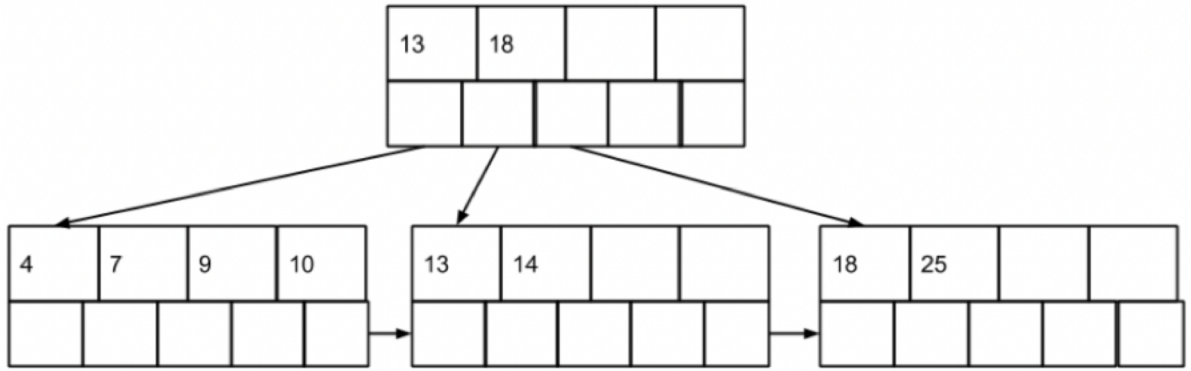


Figure 2: B+Tree

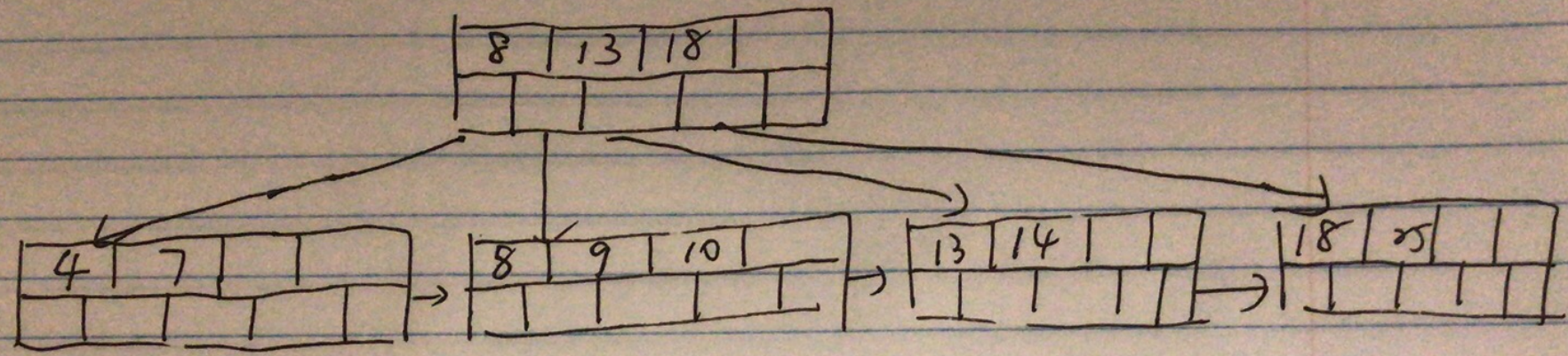
For the following two questions, **please draw the full tree**. Any incomplete drawing will results in **0** points.

(a) Show the resulting tree after inserting key 8. (*4 points*)

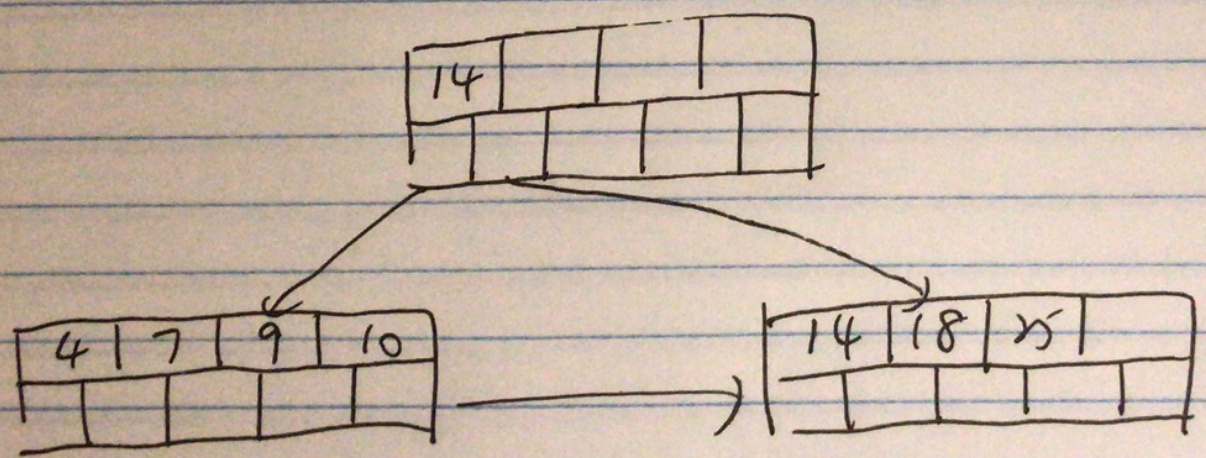


(b) Based on the original tree, show the resulting tree after deleting 13. (*4 points*)

(a)



(b)



(2) Consider indexing the following key values using an extensible hash table. Suppose that we insert the keys in the order of: 1, 15, 21, 35.
The hash function h(n) for key n is h(n) = n mod 16; i.e., the hash function is the remainder after the key value is divided by 16. Thus, the hash value is a 4-bit value. Assume that each bucket can hold 2 data items.
You have performed this indexing correctly, and have come up with the following table, as shown in figure below:

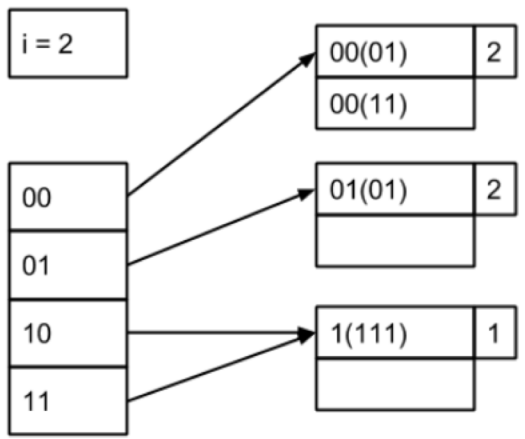
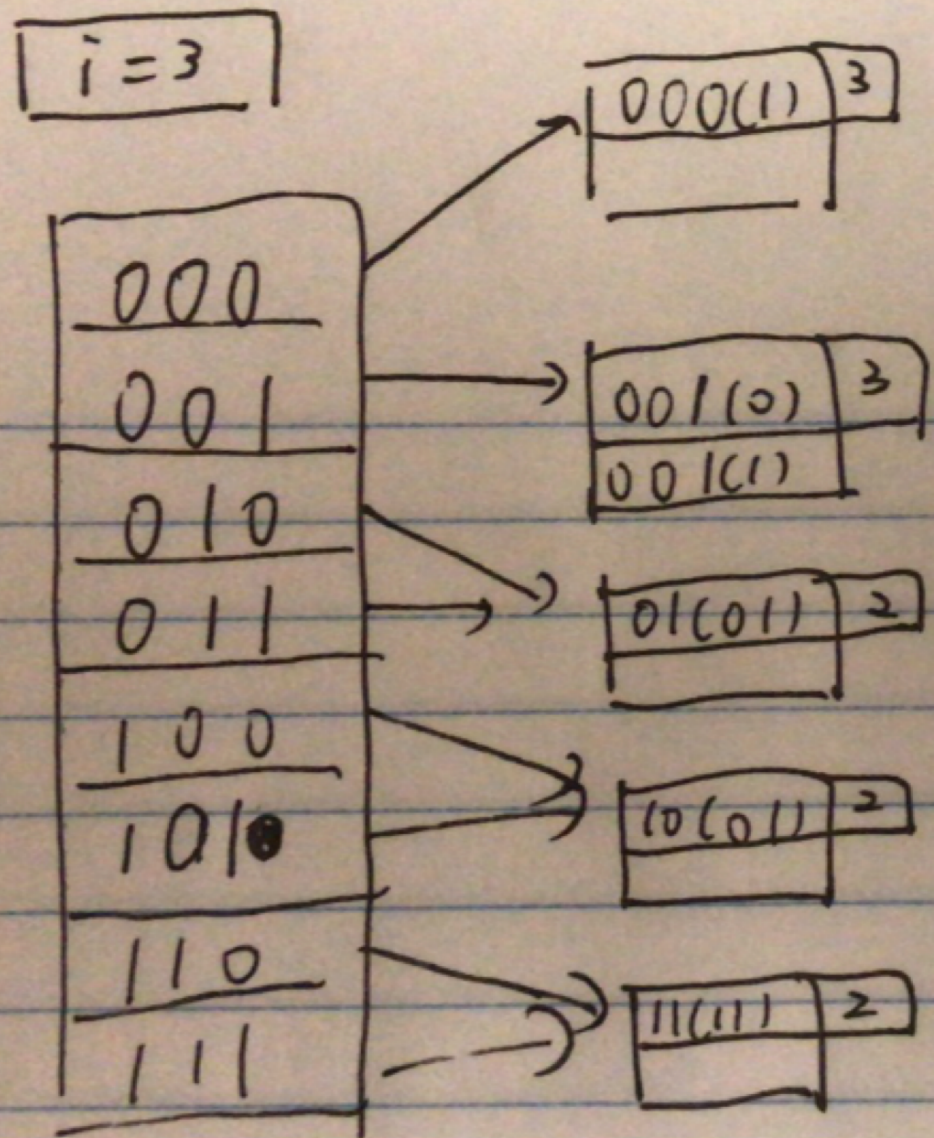


Figure 3: Extensible Hashing Table

Now insert 18 and 41, in this order, into the hash table into the table above. Redraw the table to reflect the new values. Be sure to indicate the number of bits in the hash value that are used in the array. Also, indicate the "nub" value of each block. Again, you can just draw the final table after inserting all the six keys. (*8 points*)

18 → 0010, 41 → 1001



Query Execution

Query Execution

Consider two relations, R1(A,B) and R2(B,C), with the following statistics: T(R1)=80, B(R1)=20, V(R1,A)=10, V(R1,B)=10, T(R2)=100, B(R2)=50, V(R2,B)=10, V(R1,C)=20. The memory buffer can hold 11 blocks (M=11). Assume that the data in every relation for each field is the set of integer values from 1 to the number of individual values specified for each field, evenly distributed. For instance, V(R1,B)=10 implies that the B field in relation R1 has a uniform distribution of the integer values 1 to 10.

1. What is the cost of joining R1 and R2 using a block nested-loop join? Assume that R1 is the 'outer' relation. (defined in question)

2. What is the cost of joining R1 and R2 using a partitioned hash-based join?

3. Estimate the size of the relation that is the result of:

$$\sigma_{a=8}(R1) \bowtie R2$$

Query Execution

Query Execution

Consider two relations, R1(A,B) and R2(B,C), with the following statistics: T(R1)=80, B(R1)=20, V(R1,A)=10, V(R1,B)=10, T(R2)=100, B(R2)=50, V(R2,B)=10, V(R1,C)=20. The memory buffer can hold 11 blocks (M=11). Assume that the data in every relation for each field is the set of integer values from 1 to the number of individual values specified for each field, evenly distributed. For instance, V(R1,B)=10 implies that the B field in relation R1 has a uniform distribution of the integer values 1 to 10.

1. What is the cost of joining R1 and R2 using a block nested-loop join? Assume that R1 is the 'outer' relation. (outer will be defined in the question)

   B(R1) + B(R1)B(R2)/(M-1) = 20 + 50*20/10 = 120 (Can use M or M-1, both are correct)

2. What is the cost of joining R1 and R2 using a partitioned hash-based join?

   3*(B(R1)+ B(R2)) = 3*(20 + 50) = 210

$\sigma_{a=8}(R1) \bowtie R2$

3. Estimate the size of the relation that is the result of:

   (T(R1)/V(R1,A) * T(R2)) / max(V(R1,B),V(R2,B)) = ((80/10) * 100) / 10 = 80

Problem 5 (*18 points*) Dynamic Programming

(1) Assume we have 50 tuples for relation A, i.e., $T(A) = 50$, and 30 tuples for relation B. What is the possible minimum T(A JOIN B)? Explanation is not required. (*1 points*)

(2) For the same question in **problem 5.1**, what is the possible maximum T(A JOIN B)? Explanation is not required. (*1 points*)

(3) Given relations A, B, C, D, assume $T(A) = 20$, $T(B) = 50$, $T(C) = 70$, $T(D) = 10$, and the size estimation heuristic: size(R1 JOIN R2) = 0.1 * T(R1) * T(R2). In order to find out the best query plan for **A JOIN B JOIN C JOIN D**, please fill in the following table. (*16 points*)

	Subquery	Size	Cost	Plan
1	AB			
2	AC			
3	AD			
4	BC			
5	BD			
6	CD			
7	ABC			
8	ABD			
9	ACD			
10	BCD			
11	ABCD			

(1) minimum: 0

(2) maximum: 30×50=1500

(3)

	Size	cost	Plan
AB	100	0	AB
AC	140	0	AD
AD	20	0	AC
BC	350	0	BC
BD	50	0	BD
CD	70	0	CD
ABC	700	100	(AB)C
ABD	100	20	(AD)B
ACD	140	20	(AD)C
BCD	350	50	(BD)C
ABCD	700	120	~~AB~~((AD)B)C