Practice Exam 2 KEY

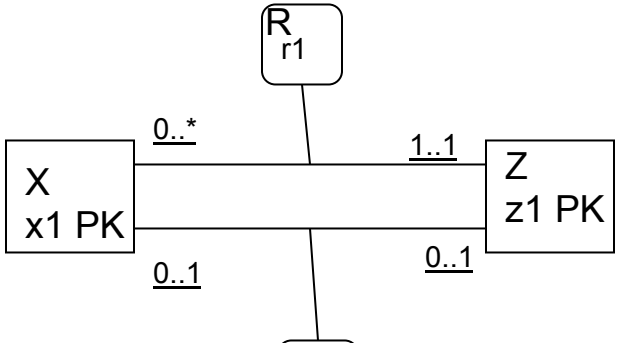
1. Consider the two tables below. Write a CREATE VIEW statement that lists the average water readings for each building of each day, but only for daily averages computed over more than 2 values. The view, call it Maintenance, should list ReadingDate, BuildingName, and the average reading for that date/building, listed as AverageValue.

```
CREATE VIEW Maintenance (ReadingDate, BuildingName, AverageValue )
AS SELECT WR.WReadingDate, WS.BuildingName,
     AVERAGE(WR.HRWReadingValue) AS AverageValue
FROM WaterSensor WS, WReading WR
WHERE WS.WaterSensorID = WR.WaterSensorID
GROUP BY WR.WReadingDate, WS.BuildingName
HAVING COUNT(*) > 2
```

```
CREATE TABLE WaterSensor (
BuildingName VARCHAR(35) NOT NULL,
WaterSensorID INTEGER,
WaterSensorOnLineDate DATE,
PRIMARY KEY (WaterSensorID));

CREATE TABLE WReading (
WaterSensorID INTEGER,
WReadingDate DATE,
WReadingTime TIME,
WValue INTEGER NOT NULL,
PRIMARY KEY (WaterSensorID, WReadingDate, WReadingTime),
FOREIGN KEY (WaterSensorID) REFERENCES WaterSensor);
```

2. (5 pts) Consider the UML fragment to the right and identify (circle) ***all*** equivalent table translations (i.e., those translations that faithfully enforce the constraints implied by the UML without regard to elegance) from those given below. You might receive partial credit for a brief explanation of your choices. UNIQUE(y) implies that y NOT NULL, but not vice versa. PK stands for PRIMARY KEY. FK stands for FOREIGN KEY.



(A)
- 0.5 pts

```
CREATE TABLE XR (
   x1, r1, z1
   PK(x1)
   FK (z1) refs ZQ
)

CREATE TABLE ZQ (
  z1, q1, x1
  UNIQUE(x1)
  PK(z1)
  FK (x1) refs XR
)
```

Z1 needs to be NOT NULL

(B)
-1 pts

```
CREATE TABLE XRQ (
  x1, r1, q1, z1
  PK(x1)
  UNIQUE(z1)
  FK (z1) refs Z
)

CREATE Z (
  z1
  PK(z1)
)
```

An X must be associated with Same Z through both R and Q

(C)
+3 pts

```
CREATE TABLE XR (
  x1, r1,
  z1 NOT NULL,
  PK(x1)
  FK (z1) refs Z
)

CREATE TABLE Z (
  z1
  PK(z1)
)

CREATE TABLE Q (
   x1, q1, z1
   PK(z1)
   UNIQUE (x1)
   FK (x1) refs XR
   FK (z1) refs Z
)
```

(D)
- 2pts

```
CREATE TABLE X (
  x1
 PK (x1)
)

CREATE TABLE R (
  x1, r1,
  z1 NOT NULL
  PK(x1)
  FK (z1) refs Z
  FK (x1) refs X
)

CREATE TABLE Z (
  z1
  PK (z1)
)

CREATE TABLE Q (
  x1, q1, z1
  PK (x1)
  UNIQUE (z1)
  FK (x1) refs X
  FK (z1) refs Z
)
```

(E)
+ 2 pts

```
CREATE TABLE X (
  x1
 PK (x1)
 FK (x1) refs R
)

CREATE TABLE R (
  x1, r1,
  z1 NOT NULL
  PK(x1)
  FK (z1) refs Z
  FK (x1) refs X
)

CREATE TABLE Z (
  z1
  PK (z1)
)

CREATE TABLE Q (
  x1, q1, z1
  PK (x1)
  UNIQUE (z1)
  FK (x1) refs X
  FK (z1) refs Z
)
```

(F) None of the above
0 points total

3. (5 pts) Consider the following table definitions:

```
CREATE TABLE RelA (Aid  integer, a1 integer, a2 integer, PRIMARY KEY (Aid))
CREATE TABLE RelB (Aid integer, Cid integer, b1 integer,
                   PRIMARY KEY (Aid, Cid, b1),
                   FOREIGN KEY (Aid) REFERENCES RelA,
                   FOREIGN KEY (Cid) REFERENCES RelC)
CREATE TABLE RelC (Cid integer, c1 integer, c2 integer, c3 integer, PRIMARY KEY (Cid))
```

Circle ***all*** queries below that are equivalent to:

```
SELECT DISTINCT C.c1
FROM  RelC C,  RelB B1,  RelA A1,  RelB B2,  RelA A2
WHERE C.Cid = B1.Cid AND B1.Aid = A1.Aid AND C.Cid = B2.Cid AND
      B2.Aid = A2.Aid AND A1.a2 = q AND A2.a2 = r
```

By equivalent, we mean produces the same output given the same input (and we are not referring to efficiency or elegance)

Total cannot exceed 5 and cannot be less than 0

(a)
```
SELECT DISTINCT C.c1
FROM  RelC C,  RelB B1,  RelA A1,  RelB B2,  RelA A2
WHERE A1.a2 = q AND A2.a2 = r AND
      C.Cid = B1.Cid AND B1.Aid = A1.Aid AND
      C.Cid = B2.Cid AND B2.Aid = A2.Aid
```

+1 pt

(b)
```
SELECT DISTINCT C.c1
FROM RelA A, RelB B, RelC C
WHERE C.Cid = B.Cid AND B.Aid = A.Aid AND A.a2 = q AND
      C.Cid IN (SELECT C2.Cid
                FROM RelC C2, RelA A2, RelB B2
                WHERE C2.Cid = B2.Cid AND
                      B2.Aid = A2.Aid  AND A2.a2 = r )
```

+2 pts

(c)
```
SELECT DISTINCT C.c1
FROM RelC C, RelB B, RelA A
WHERE C.Cid = B.Cid AND B.Aid = A.Aid AND A.a2 = q
INTERSECT
SELECT DISTINCT C2.c1
FROM RelC C2, RelB B2, RelA A2
WHERE C2.Cid = B2.Cid AND B2.Aid = A2.Aid AND A2.a2 = r
```

-1 pt

(d)
```
SELECT DISTINCT C.c1
FROM RelC C
WHERE C.Cid IN (( SELECT B.Cid
                  FROM RelA A, RelB B
                  WHERE B.Aid = A.Aid AND A.a2 = q)
                INTERSECT
                (( SELECT B2.Cid
                  FROM RelA A2, RelB B2
                  WHERE B2.Aid = A2.Aid AND A2.a2 = r))
```

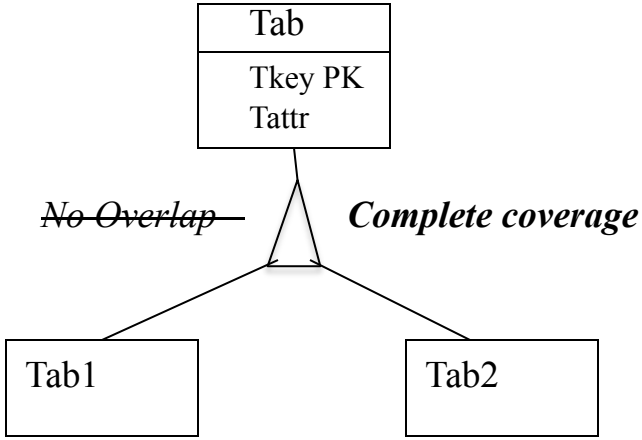
+2 pts

(e) None of the above

0 pts total

4. ~~Circle all options that would correctly enforce the No Overlap constraint between Tab1 and Tab2 in an SQL translation of the following ER fragment.~~

Design a question like this, but for translating the COMPLETE COVERAGE constraint!



a) ~~CREATE ASSERTION NoOverlapBetweenTab1AndTab2~~
~~CHECK (NOT EXISTS (SELECT * FROM Tab1 T1, Tab2 T2 WHERE T1.Tkey = T2.Tkey))~~

b) ~~CREATE ASSERTION NoOverlapBetweenTab1AndTab2~~
~~CHECK (NOT EXISTS (SELECT Tkey FROM Tab1) INTERSECT (SELECT Tkey FROM Tab2))~~

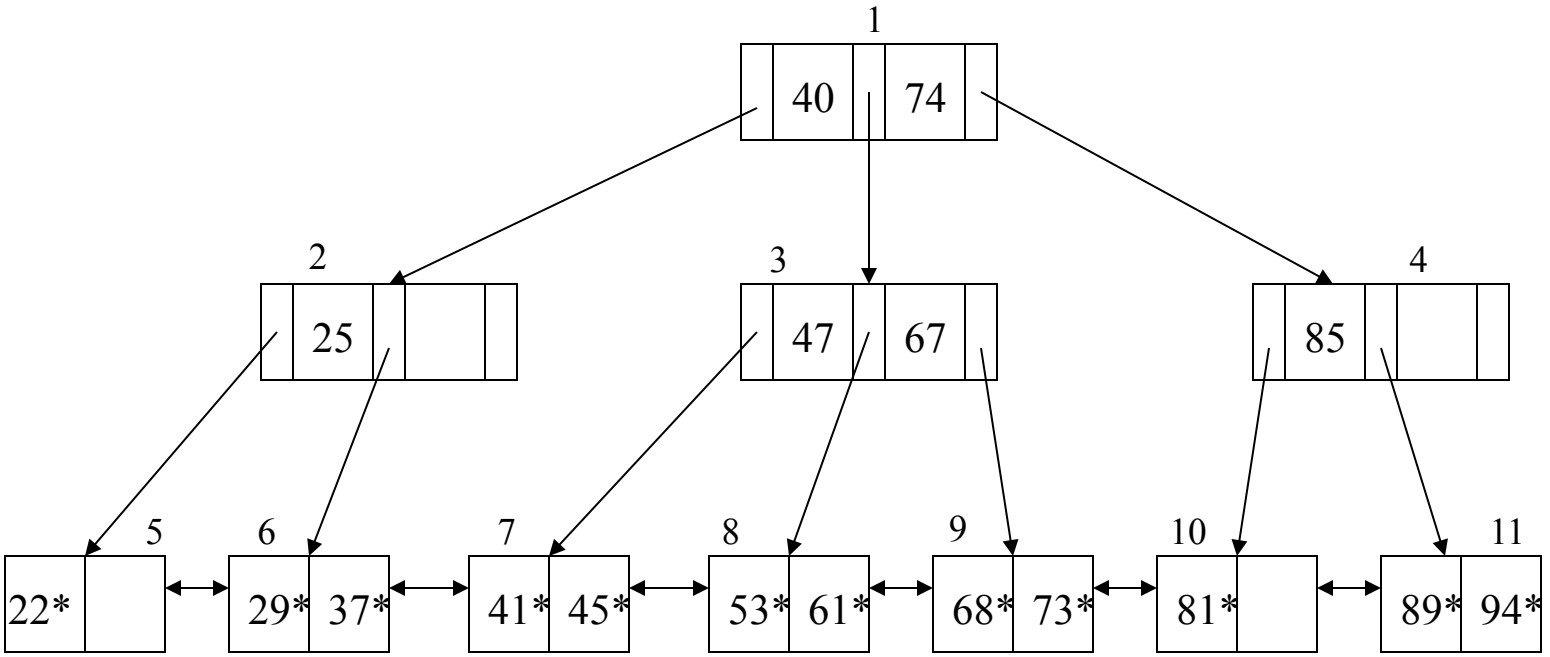
c) ~~CREATE ASSERTION NoOverlapBetweenTab1AndTab2~~
~~CHECK ((SELECT COUNT (Tab1.Tkey) FROM Tab1) = (SELECT COUNT (Tab2.Tkey) FROM Tab2))~~

d) ~~CREATE ASSERTION NoOverlapBetweenTab1AndTab2~~
~~CHECK ((SELECT COUNT (Tab1.Tattr) FROM Tab1) = (SELECT COUNT (Tab2.Tattr) FROM Tab2))~~

e) ~~CREATE ASSERTION NoOverlapBetweenTab1AndTab2~~
~~CHECK (EXISTS (SELECT * FROM Tab1 T1, Tab2 T2 WHERE T1.Tkey = T2.Tkey))~~

f) None of the above

5. (5 pts) Consider the B+ tree index for attribute A of table T. Above each node is a numeric label for the node (1 through 11), which you will use in answering this question.



For each of the following query, update, and insert operations, list the *nodes* (by label) that would have to be changed (e.g., split, new values added) at some point in performing the respective operation. You will not be graded on the order in which you list the nodes. If no nodes need be changed, the write *None*. Ignore data nodes, which are not shown, and do not list new nodes that might be introduced. Assume that this index for attribute A is used in evaluating each operation below. Treat each operation as independent – these represent alternative actions on the tree above, not a sequence of actions.

(a) SELECT T.A FROM T WHERE T.A > 29 : None 1 pt, all or nothing

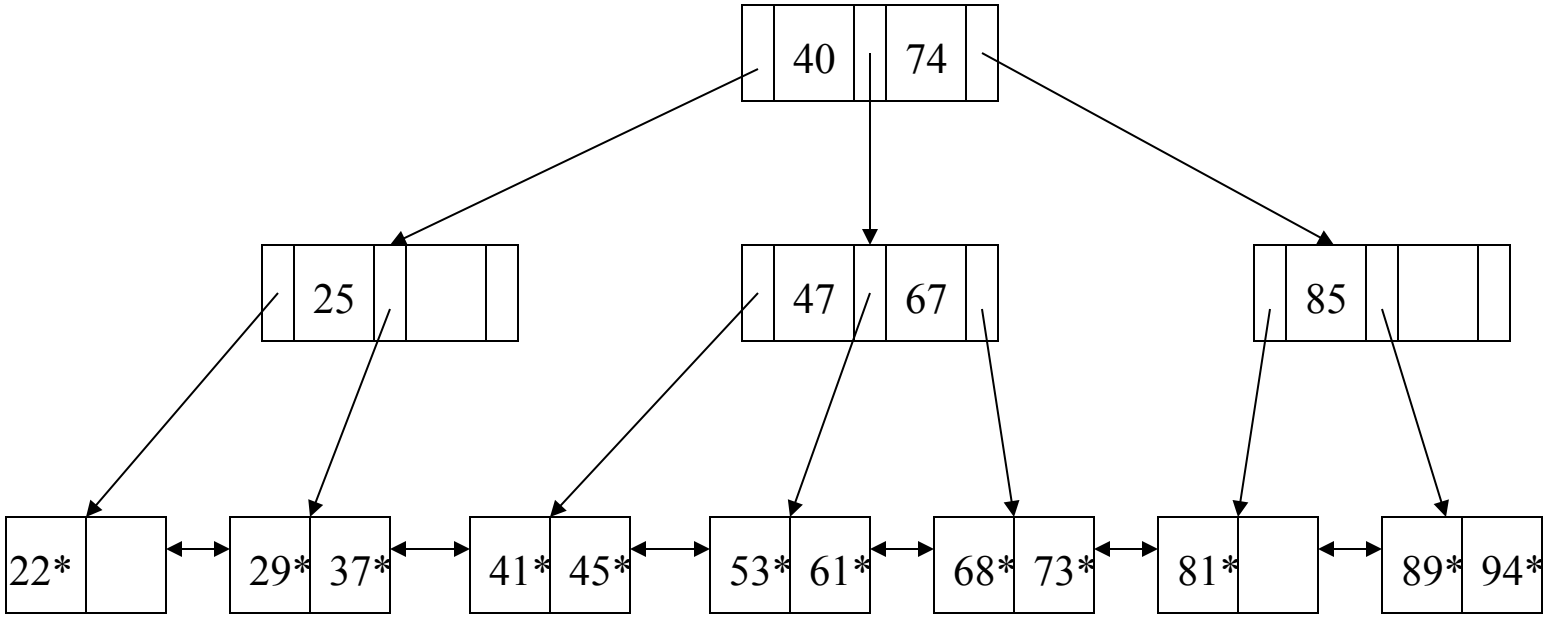
(b) UPDATE T SET T.B = T.B + 100 WHERE T.A = 53 : None 1 pt, all or nothing

(c) UPDATE T SET T.A = T.A + 100 WHERE T.A = 53 : 8, 11, 4 -0.5 for each unlisted; -0.5 for each extra [0-1.5]

(d) INSERT INTO T (A, B, C) VALUES (56, 47, 32) : 8, 3, 1 -0.5 for each unlisted; -0.5 for each extra [0-1.5]

-0.5 for each uncircled node below or extra circled node (up to –2)

6. (3 pts) Consider the B+ tree below.

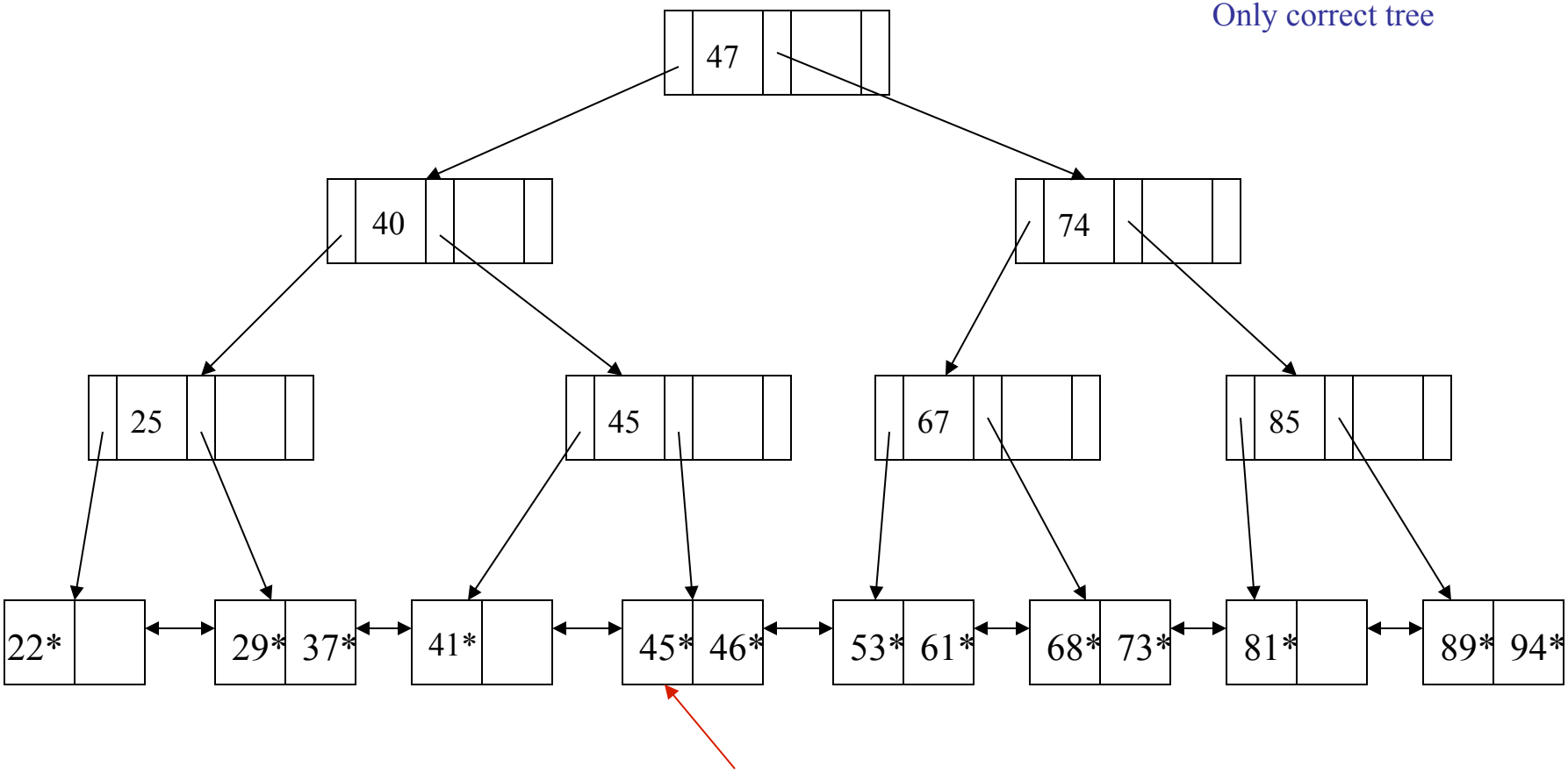


Note that this tree does not show data nodes, and you do not need to see the data nodes to answer this question. At each leaf, N* is an index of the form <N, <page id, slot #>>, where N is the value of the search key.

Show the tree that results from inserting a record with search key **46** (assuming **no** redistribution).

Write resulting B+ tree for (b) here

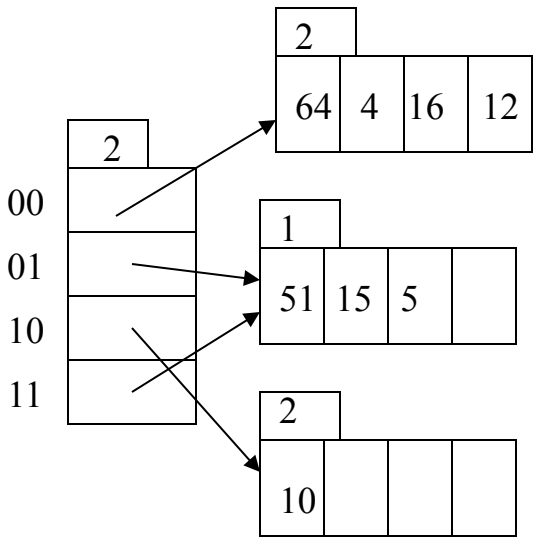
Only correct tree

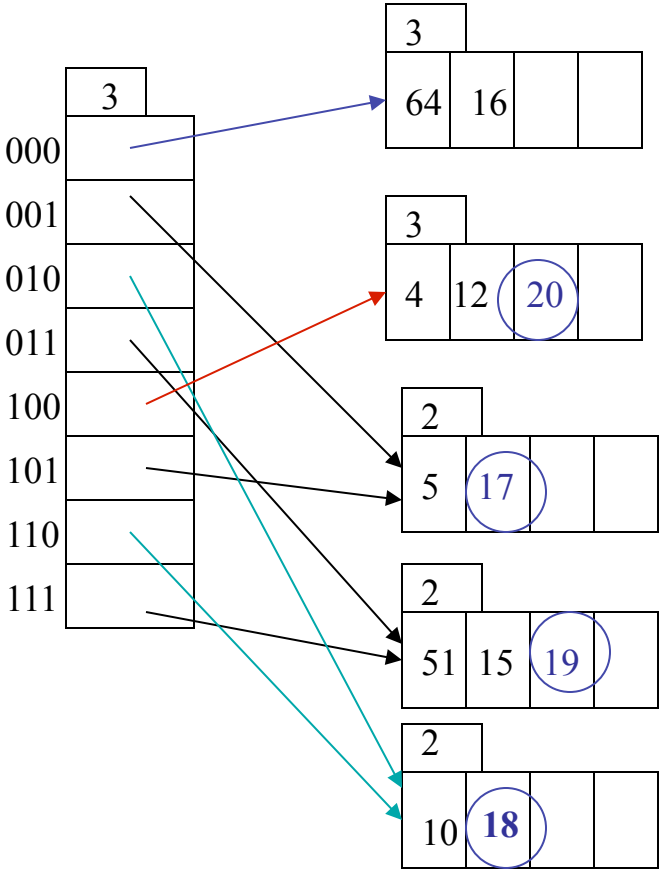


-2 if 45 does not appear at leaf

-3 for any tree that isn't 4 levels. Use discretion on partial credit.

7. (5 pts) Consider the extendible hash table to the left. Assume Hash(x) = x. Show the result of inserting the following keys in order: 18, 17, 20, 19

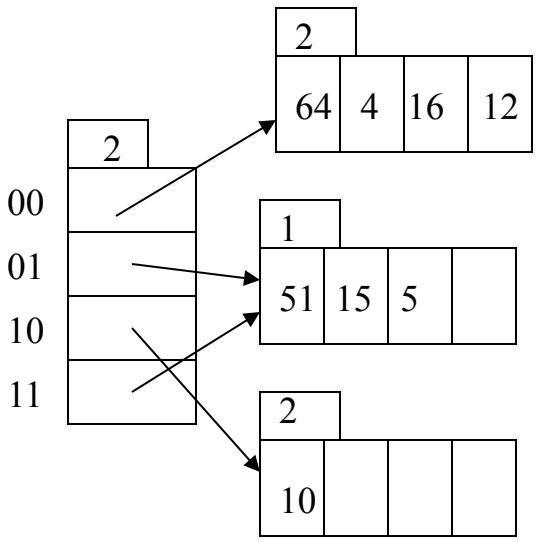




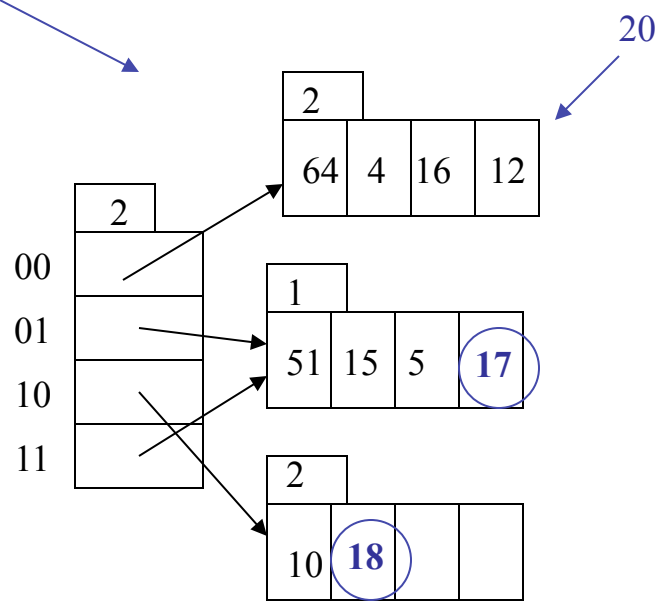
-1 for each misplaced key (order within a bin not important)

Answer here

7. (5 pts) Consider the extendible hash table to the left. Assume Hash(x) = x. Show the result of inserting the following keys in order: 18, 17, 20, 19

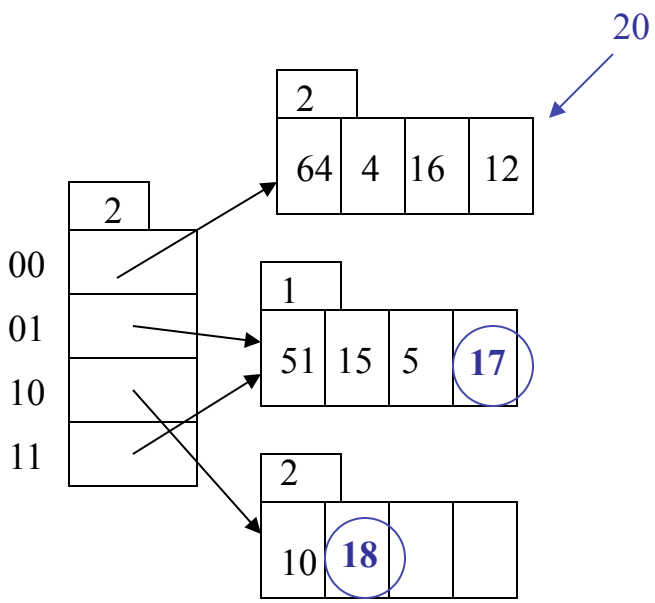


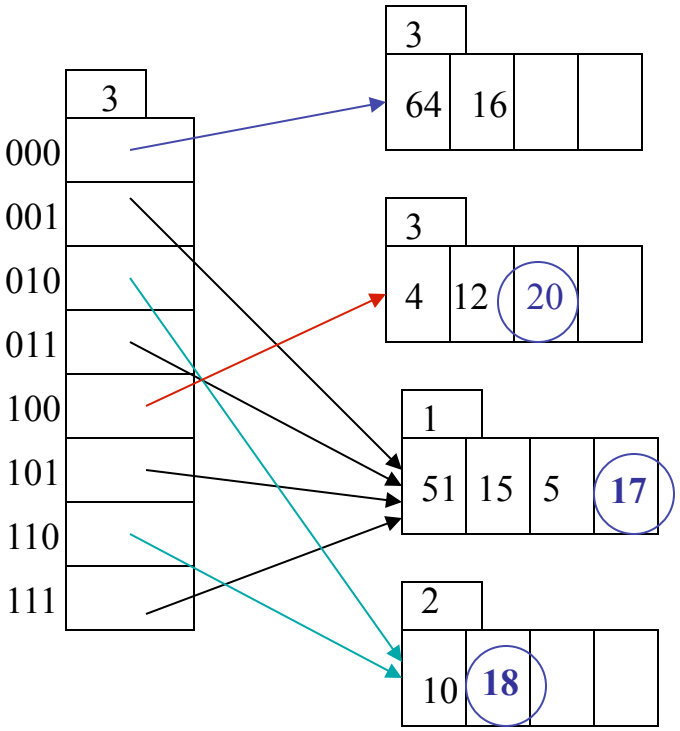
Intermediate answer



Answer here

7. (5 pts) Consider the extendible hash table to the left. Assume Hash(x) = x. Show the result of inserting the following keys in order: ~~18~~, ~~17~~, 20, 19

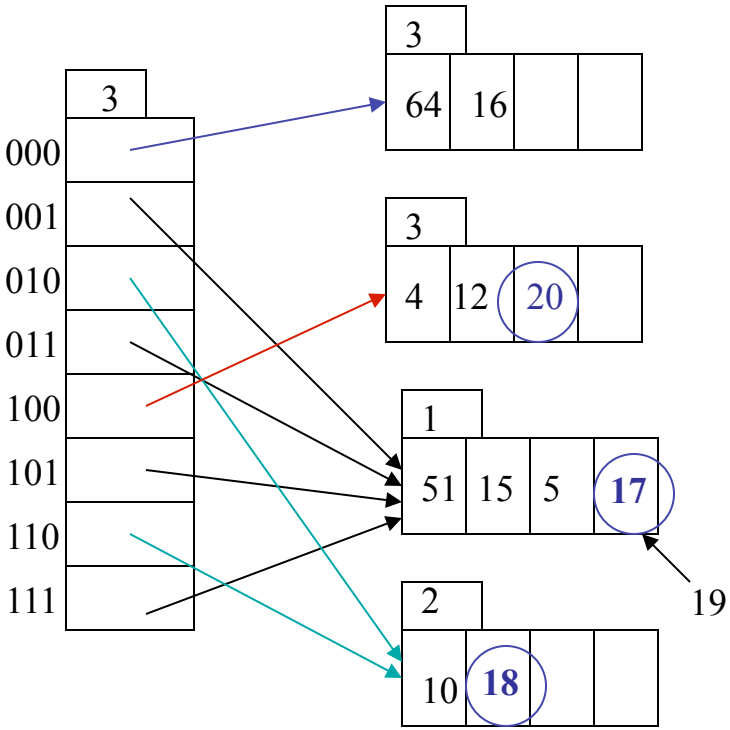


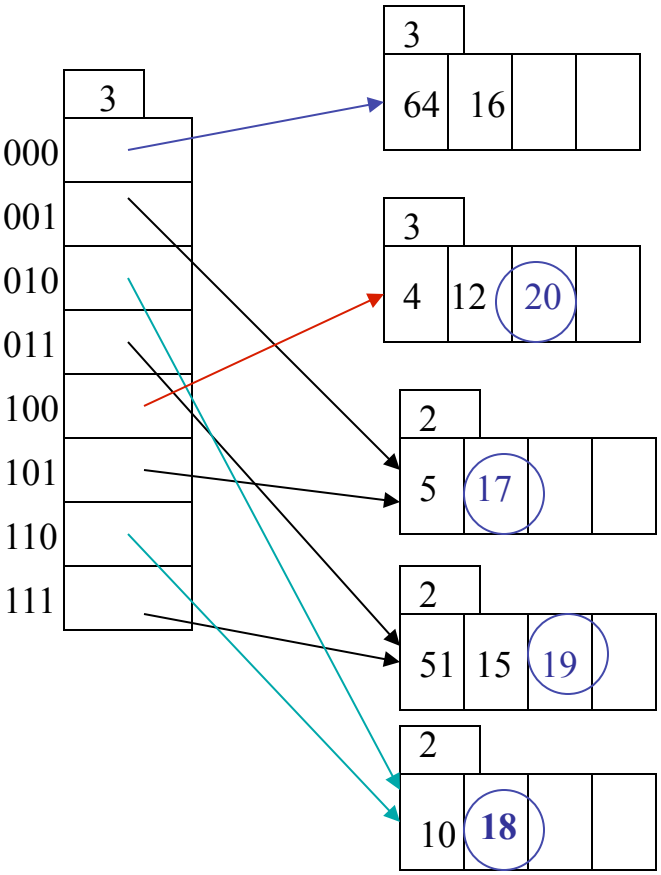


Intermediate answer

Answer here

7. (5 pts) Consider the extendible hash table to the left. Assume Hash(x) = x. Show the result of inserting the following keys in order: ~~18~~, ~~17~~, ~~20~~, 19



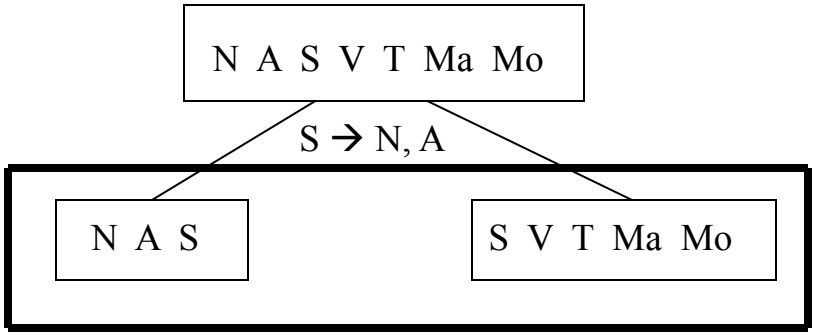


Final answer

Answer here

8. (4 pts) Consider the following FDs and the decomposition into <u>two</u> tables based on them below.

S → N, A
V → Mo, S
Mo → Ma, T



a) (1 pt) Is the decomposition lossless or lossy (not lossless)?

Lossless (1 pt)

b) (1 pt) Is the decomposition dependency preserving or not?

Yes, dependency preserving (1 pt)

c) (1 pt) For each of the two tables in the decomposition, indicate whether it is in BCNF?

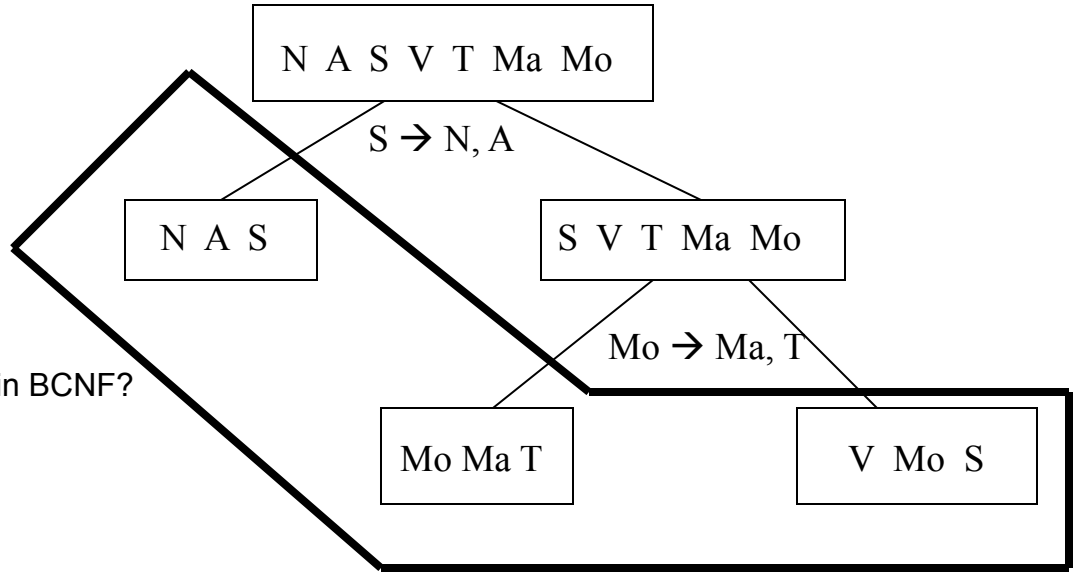
The left table is in BCNF. (0.5 pts) The right table is NOT in BCNF. (0.5 pts)

d) (1 pt) List at least one reason why we might prefer this two-table decomposition over the three-table decomposition of the next question.

The two-table decomposition does not require some of the joining that is required of the next question's decomposition, thus saving time for queries involving the attributes S V T Ma Mo

9. (4 pts) Consider the following FDs and the lossless, dependency-preserving decomposition into <u>three</u> tables based on them below.

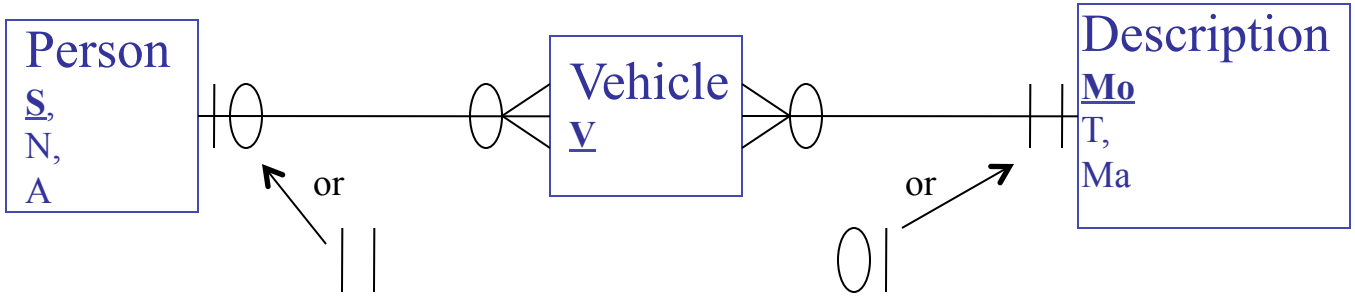
S → N, A
V → Mo, S
Mo → Ma, T



a) (1 pt) For each of the tables in the decomposition, indicate whether it is in BCNF?

All in BCNF (all or nothing)

b) (3 pts) Draw an ER diagram that would translate into the three tables of the decomposition.



-2 if missing one or more cardinality constraints;

10. (3 pts) Consider the relational schema C S J D P Q V K with FDs JP→C, SD→P, J→S, C→SJDPQV

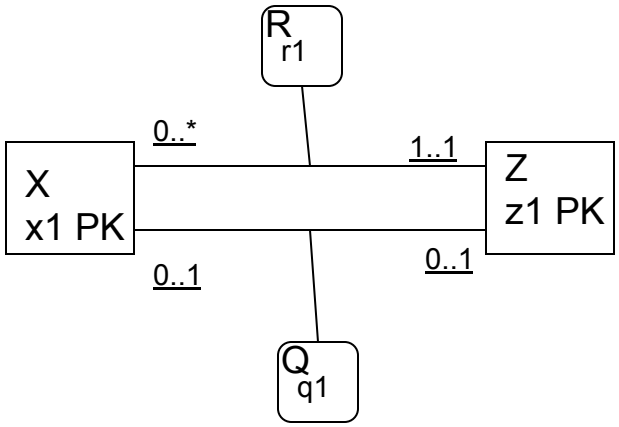
Give *all* keys for this relational schema.

JPK, CK, JDK

1 pt for each, -1 for any others

11. A bank's database needs to store information about **employees** (keyed by **SSN**, with additional attributes of **name**, **salary**, and **phone**); **branches** of the bank (keyed by **bno**, with **bname** and **address** as additional attributes); and **customers** (keyed by **SSN**, with additional attributes **name** and **address)**. Each **employee** is assigned to exactly one **branch** of the bank, with the start date at that branch recorded. Each **employee** is managed by at most one other **employee**. A **customer** has some combination of one or more **loans** from the bank and/or one or more deposit **accounts** with the bank. Each **loan** is identified by a **loan-number** with a record of the current **balance**. A complete history of loan payments is recorded giving the **payment date** and **payment amount** for each payment on the loan. There are never two payments for the same loan recorded for the same date. Each **loan** is administered by exactly one **branch** of the bank. An **account** is identified by an **account-number**, with an additional indication of the **current balance**. An **account** may be a checking account (with an allowable **overdraft** amount recorded) or a savings account (with an **interest rate** recorded). Draw an ER diagram that captures this information.

WILL POST KEY Wednesday morning – try to put this together yourself


R
r1
0..*
1..1
X
x1 PK
Z
z1 PK
0..1
0..1
Q
q1