Name	CWID

Quiz 1

Sept 22th, 2020
Due Sept 30st, 11:59pm

Quiz 1: CS525 - Advanced Database Organization

Results

Please leave this empty!

1.1	1.2	1.3	1.4	Sum

Instructions

- **You have to hand in the assignment using your blackboard**
- **This is an individual and not a group assignment**
- Multiple choice questions are graded in the following way: You get points for correct answers and points subtracted for wrong answers. The minimum points for each questions is **0**. For example, assume there is a multiple choice question with 6 answers - each may be correct or incorrect - and each answer gives 1 point. If you answer 3 questions correct and 3 incorrect you get 0 points. If you answer 4 questions correct and 2 incorrect you get 2 points. . . .
- For your convenience the number of points for each part and questions are shown in parenthesis.
- There are 4 parts in this quiz
  1. SQL
  2. Relational Algebra
  3. Index Structures
  4. Result Size Estimation

Part 1.1 SQL (Total: 31 + 10 bonus points Points)

Consider the following publication schema and example instance. The example data should not be used to formulate queries. SQL statements that you write should return the correct result for every possible instance of the schema!

Author

aname	affiliation
Peter Smith	University of Science
Anish Kapoor	UNC
Yutang Li	UNC

AuthorPublication

aname	title
Peter Smith	A Mole of moles
Peter Smith	A tractus on logic
Anish Kapoor	Analyzing the frequency of studies: a meta review
Yutang Li	Analyzing the frequency of studies: a meta review

Article

title	journal	number	issue
A Mole of moles	Journal of P-Hacking	1	12
A tractus on logic	Journal of Bogus Research	3	1
Analyzing the frequency of studies: a meta review	Science	4	3

Journal

jname	field	pagelimit	publisher
VLDB Journal	CS	20	Springer
Science	All	12	Springer
Journal of Bogus Research	All	8	Heist
Journal of P-Hacking	Statistics	12	Heist

Publisher

pname	location	type
Springer	Berlin	Academic
Heist	Bern	Predatory
EST	Charlotte	Academic

Hints:

- Attributes with black background are the primary key attributes of a relation
- Attribute `aname` of relation `AuthorPublication` is a foreign keys to relation `Author`.
- Attribute `title` of relation `AuthorPublication` is a foreign keys to relation `Article`.
- Attribute `journal` of relation `Article` is a foreign keys to relation `Journal`.
- Attribute `publisher` of relation `publisher` is a foreign keys to relation `Publisher`.

Question 1.1.1 (3 Points)

Write an SQL query that returns the names of researchers from the university of science that have exclusively published in predatory journals (`type` is “*Predatory*”).

Solution

```
SELECT aname
FROM Author a
WHERE affiliation = 'University␣of␣Science'
      AND NOT EXISTS (
          SELECT *
          FROM AuthorPublication ap, Article c, Journal j, Publisher p
          WHERE a.aname = ap.aname
                AND ap.title = c.title
                AND c.journal = j.jname
                AND j.publisher = p.publisher
                AND p.type <> 'Predatory'
      )
```

Question 1.1.2 (5 Points)

Write an SQL query that returns the names of authors that have published in the same journals, i.e., a pair of authors *author1* and *author2* should be return if the set of journals *author1* published in is equal to the set of journals *author2* published in.

Solution

```
WITH author_journal AS (
  SELECT DISTINCT a.aname, r.journal
  FROM Author a, AuthorPublication ap, Article r
  WHERE a.aname = ap.aname AND ap.title = r.title
)
SELECT aj1.aname AS author1, aj2.aname AS author2
FROM author_journal aj1,
     author_journal aj2
WHERE aj1.aname <> aj2.aname
      AND NOT EXISTS (SELECT *
                      FROM author_journal aj3
                      WHERE aj3.aname = aj1.aname
                            AND aj3.journal NOT IN (
                                SELECT aj4.journal
                                FROM author_journal aj4
                                WHERE aj4.aname = aj2.name
                        ))
      AND NOT EXISTS (SELECT *
                       FROM author_journal aj3
                       WHERE aj3.aname = aj2.aname
                             AND aj3.journal NOT IN (
                                 SELECT aj4.journal
                                 FROM author_journal aj4
                                 WHERE aj4.aname = aj1.name
                        ))
```

Question 1.1.3 (5 Points)

Write a query that returns the journal with the highest growth factor averaged across all of the issues of the journal. The growth factor of issue x of a journal is defined as the number of articles published in this issue divided by the number of articles published in issue $x - 1$ (this is only defined for $x > 1$).

Solution

```
WITH growth_factor AS (
  SELECT num_pub / LAG(num_pub,1) OVER (PARTITION BY journal ORDER BY issue) AS gf,
         journal
  FROM  (SELECT count(*) AS num_pub, journal, issue
         FROM Article a1
         GROUP BY journal, issue)
  WHERE issue > (SELECT min(issue) FROM Article a2 WHERE a1.journal = a2.journal)
)
WITH avg_growth AS (
  SELECT avg(gf) AS avg_gf, journal
  FROM growth_factor
  GROUP BY journal
)
SELECT journal
FROM avg_growth
WHERE avg_gf = (SELECT max(avg_gf) FROM avg_growth);
```

Question 1.1.4 (2 Points)

Return the names of the 3 researchers that have published the most articles.

Solution

```
SELECT aname
FROM (SELECT count(*) AS num_publ, aname
      FROM AuthorPublication
      GROUP BY aname) publcnt
ORDER BY num_publ DESC
LIMIT 3
```

Question 1.1.5 (2 Points)

Write an SQL query that returns publishers that publish more than 5 journals.

Solution

```
SELECT publisher
FROM Journal
GROUP BY publisher
HAVING count(*) > 5;
```

Question 1.1.6 (3 Points)

Write an SQL query that returns pairs of authors (a_1, a_2) that are direct co-author (have published an article together) or indirect co-authors (authors a_1 has published with a third author a_3 who is a direct or indirect co-authors of the other author a_2).

Solution

```
WITH RECURSIVE tc AS (
  SELECT a1.aname AS a1, a2.aname AS a2
  FROM AuthorPublication a1, AuthorPublication a2
  WHERE a1.title = a2.title
  UNION -- recursion
  SELECT a1, a3.aname AS a2
  FROM tc, AuthorPublication a2, AuthorPublication a3
  WHERE a2 = a2.aname AND a2.title = a3.title
)
SELECT * FROM tc;
```

Question 1.1.7 (5 Points)

Write an SQL query that returns pairs of authors that are from the same university and have published in the same set of fields.

Solution

```
WITH same_uni AS  (
  SELECT a1.aname AS a1, a2.aname AS a2
  FROM Author a1, Author a2
  WHERE a1.affiliation = a2.affiliation
)
author_fields AS (
  SELECT DISTINCT field, a.aname
  FROM Author a, AuthorPublication ap, Article r,  Journal j
  WHERE a.aname = ap.aname AND ap.title = r.title  AND r.journal = j.jname
)
afield_cnt AS (
  SELECT count(*) nfields, aname
  FROM  author_fields
  GROUP BY aname
)
apair_field_cnt AS (
  SELECT count(*) AS nfields, a1, a2
  FROM same_uni, author_fields f1, author_fields f2
  WHERE a1 = f1.aname AND a2 = f2.aname AND f1.field = f2.field
  GROUP BY a1, a2
)
SELECT a1, a2
FROM apair_field_cnt p, afield_cnt f1, afield_cnt f2
WHERE p.nfields = f1.nfields
      AND p.nfields = f2.nfields
      AND a1 = f1.aname
      AND a2 = f2.aname
```

Question 1.1.8 (2 Points)

Write an SQL query that returns the accumulative count of published articles for each journal over its issues.

Solution

```
SELECT DISTINCT count(*) OVER (PARTITION BY journal
                               ORDER BY issue) AS num_publ,
                journal,
                issue
FROM Article
```

Question 1.1.9 (4 Points)

Write an SQL query that returns names of journal which have seen a significant drop (50%) in the number of published articles in the last three issues. That is, for each issue you have to count the number of published articles in that issue and the two preceding issues. The drop is then determined by comparing the numbers of two adjacent issues x and $x + 1$.

Solution

```
WITH last_three AS (
  SELECT DISTINCT journal,
                  issue,
                  count(*) OVER (PARTITION BY journal
                                 ORDER BY issue
                                 ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) AS cnt
  FROM Article a
)
SELECT DISTINCT journal
FROM (SELECT journal,
             cnt,
             LAG(cnt, 1) OVER (PARTITION BY journal ORDER BY issue) AS prev_cnt
      FROM last_three) lag
WHERE cnt < 0.5 * prev_cnt;
```

Question 1.1.10 Optional Bonus Question (10 Bonus Points)

Consider a relation storing information about items in a warehouse. The relation stores for each item its type and the time interval during which the item was stored in the warehouse. An example instance is shown below. Write a single SQL query that returns for each time interval how many items are in the warehouse. Note that there is some freedom in how the result is encoded. For instance, two possible equivalent query results are shown below. **Note: The result returned by the query should not contain any overlapping time intervals!**

warehouse

item	from	to
Sausage	2020-01-01	2020-01-03
Eggs	2020-01-01	2020-01-02
Bacon	2020-01-03	2020-01-03

query result (one option)

cnt	from	to
2	2020-01-01	2020-01-03

query result (another option)

cnt	from	to
2	2020-01-01	2020-01-02
2	2020-01-03	2020-01-03

Solution

```
WITH endpoints AS (
  SELECT from AS pnt, 0 AS isend, 1 AS cnt FROM warehouse
  UNION ALL
  SELECT to + 1, 1 AS isend, -1 AS cnt FROM warehouse
),
preagg AS (
  SELECT pnt, isend, sum(cnt) AS cnt
  FROM endpoints
  GROUP BY pnt, isend
)
SELECT cnt, start, end
FROM (SELECT pnt AS start, LEAD(pnt,1) OVER (ORDER BY pnt) AS end, cnt
      FROM (SELECT pnt, isend, sum(cnt) OVER (ORDER BY pnt, isend) AS cnt
            FROM warehouse)
WHERE end IS NOT NULL;
```

Part 1.2 Relational Algebra (Total: 29 Points)

Question 1.2.1 Relational Algebra (4 Points)

Write a relational (bag semantics) algebra expression over the schema from the SQL part (part 1) that returns universities for which at least 3 authors have published in the Science journal.

Solution

$$q_{cscience} = {}_{affiliation}\alpha_{count(*)}(Author \bowtie AuthorPublication \bowtie \sigma_{journal=Science}(Article))$$
$$q = \pi_{affiliation}(\sigma_{count(*)\geq 3}(q_{cscience}))$$

Question 1.2.2 Relational Algebra (3 Points)

Write a relational (bag semantics) algebra expression over the schema from the SQL part (part 1) that returns the number of authors for each paper.

Solution

$${}_{count(*)}\alpha_{title}(AuthorPublication)$$

Question 1.2.3 Relational Algebra (5 Points)

Write a relational (bag semantics) algebra expression over the schema from the SQL part (part 1) that returns authors that have published in all journals.

Solution

$$\begin{aligned}
q_{posCombinations} &= \pi_{aname}(Author) \times \pi_{jname}(Journal)\\
q_{existCombinations} &= \pi_{aname,journal}(AuthorPublication \bowtie Article)\\
q_{notAllJournals} &= \pi_{aname}(q_{posCombinations} - q_{existsCombinations})\\
q &= \pi_{aname}(Author) - q_{notAllJournals}
\end{aligned}$$

Question 1.2.4 SQL → Relational Algebra (5 Points)

Translate the SQL query from Question 1.1.1 into relational algebra (bag semantics).

Solution

$$\begin{aligned}
q_{nonPred} &= \pi_{aname}(AuthorPublication \bowtie Article \bowtie_{journal=jname} Journal \bowtie_{publisher=pname} \sigma_{type \neq Predatory}(Publisher))\\
q &= \pi_{aname}(\sigma_{affiliation=UniversityofScience}(Author)) - q_{nonPred}
\end{aligned}$$

Question 1.2.5 SQL → Relational Algebra (6 Points)

Translate the SQL query from question 1.1.2 into relational algebra (bag semantics).

Solution

$$\begin{aligned}
q_{authjour} &= \delta(\pi_{aname,journal}(Author \bowtie AuthorPublication \bowtie Article)\\
q_{authorPairs} &= \sigma_{a1 \neq a2}(\pi_{aname \rightarrow a1}(Author) \times \pi_{aname \rightarrow a2}(Author))\\
q_{posCombinations} &= \pi_{aname \rightarrow a1,journal}(q_{authjour}) \times \pi_{aname \rightarrow a2}(Author)\\
q_{misComb} &= \pi_{a1,a2}(q_{posCombinations} - \pi_{a1,a2,journal}(\pi_{aname \rightarrow a1,journal}(q_{authjour}) \bowtie \pi_{aname \rightarrow a2,journal}(q_{authjour})))\\
q &= q_{authorPairs} - q_{misComb} - \pi_{a2,a1}(q_{misComb})
\end{aligned}$$

Question 1.2.6 SQL → Relational Algebra (6 Points)

Translate the SQL query from question 1.1.4 into relational algebra (bag semantics).

Solution

$$\begin{aligned}
q_{cnt} &= {}_{count(*)\rightarrow cnt}\alpha_{aname}(AuthorPublication)\\
q_{max} &= {}_{max(cnt)\rightarrow mcnt}\alpha(q_{cnt}) \bowtie_{mcnt=cnt} q_{cnt}\\
q_{snd} &= {}_{max(cnt)\rightarrow mcnt}\alpha(\pi_{cnt}(q_{cnt}) - \pi_{mcnt}(q_{max})) \bowtie_{mcnt=cnt} q_{cnt}\\
q_{third} &= {}_{max(cnt)\rightarrow mcnt}\alpha(\pi_{cnt}(q_{cnt}) - \pi_{mcnt}(q_{max}) - \pi_{mcnt}(q_{snd})) \bowtie_{mcnt=cnt} q_{cnt}\\
q &= \pi_{aname}(q_{max}) \cup \pi_{aname}(q_{snd}) \cup \pi_{aname}(q_{third})
\end{aligned}$$

Question 1.2.7 Equivalences (4 Points)

Consider the following relation schemas (primary key attributes are underlined):

$R(\underline{A}, B)$, $S(\underline{B}, C)$, $T(\underline{C}, D)$, $U(\underline{E}, F, G)$.

Furthermore, assume that $R.B$ is a foreign key to S and $S.C$ is a foreign key to T. Check equivalences that are correct under **bag semantics**. For example $\delta(R \bowtie R) \equiv R$ should be checked, whereas $R \equiv S$ should not be checked.

[✓] $\delta(\delta(R)) \equiv \delta(R)$

[✓] ${}_{F}\alpha_{sum(E)}(U) \equiv {}_{F}\alpha_{sum(X)}({}_{F,G}\alpha_{sum(E)\rightarrow X}(U))$

☐ $\delta(\pi_A(R \bowtie_{B=C} S)) \equiv \pi_A(R \bowtie_{B=C} S)$

[✓] $R \triangleright S \equiv R - (\pi_{A,B}(R \bowtie S))$

[✓] $\pi_B(R \bowtie S) \equiv \pi_B(R)$

☐ $\pi_C(S \bowtie R) \equiv \pi_C(S)$

[✓] $\delta(\pi_A(R)) \equiv \pi_A(R)$

☐ $\pi_A(R \bowtie S) \equiv \pi_A(R)$

Part 1.3 Index Structures (Total: 30 Points)

Assume that you have the following table:

Item

SSN	name	age
11	Pete	13
24	Bob	16
25	John	55
26	Joe	44
28	Alice	33
32	Lily	65
34	Gertrud	71
39	Heinz	12

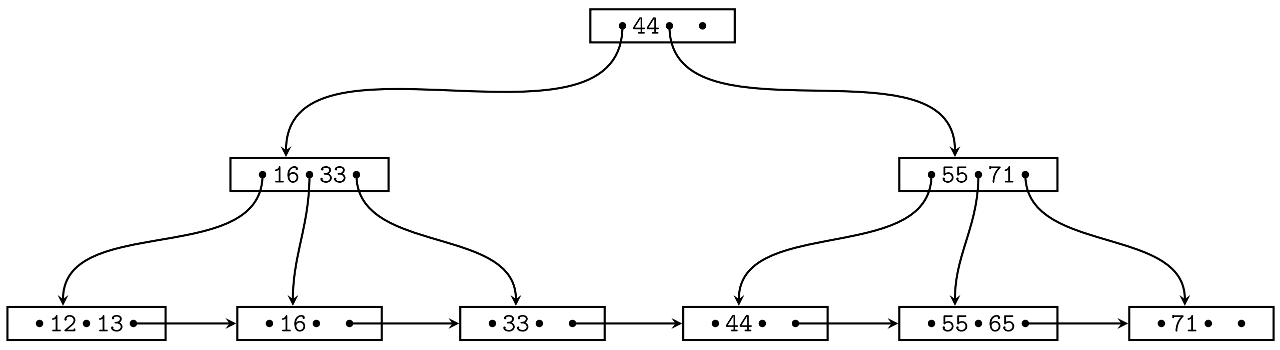
Question 1.3.1 Construction (12 Points)

Create a B+-tree for table *Item* on key *age* with $n = 2$ (up to two keys per node). You should start with an empty B+-tree and insert the keys in the order shown in the table above. Write down the resulting B+-tree after each step.

When splitting or merging nodes follow these conventions:

- **Leaf Split**: In case a leaf node needs to be split during insertion and n is even, the left node should get the extra key. E.g, if $n = 2$ and we insert a key 4 into a node [1,5], then the resulting nodes should be [1,4] and [5]. For odd values of n we can always evenly split the keys between the two nodes. In both cases the value inserted into the parent is the smallest value of the right node.
- **Non-Leaf Split**: In case a non-leaf node needs to be split and n is odd, we cannot split the node evenly (one of the new nodes will have one more key). In this case the "middle" value inserted into the parent should be taken from the right node. E.g., if $n = 3$ and we have to split a non-leaf node [1,3,4,5], the resulting nodes would be [1,3] and [5]. The value inserted into the parent would be 4.
- **Node Underflow**: In case of a node underflow you should first try to redistribute values from a sibling and only if this fails merge the node with one of its siblings. Both approaches should prefer the left sibling. E.g., if we can borrow values from both the left and right sibling, you should borrow from the left one.

Solution

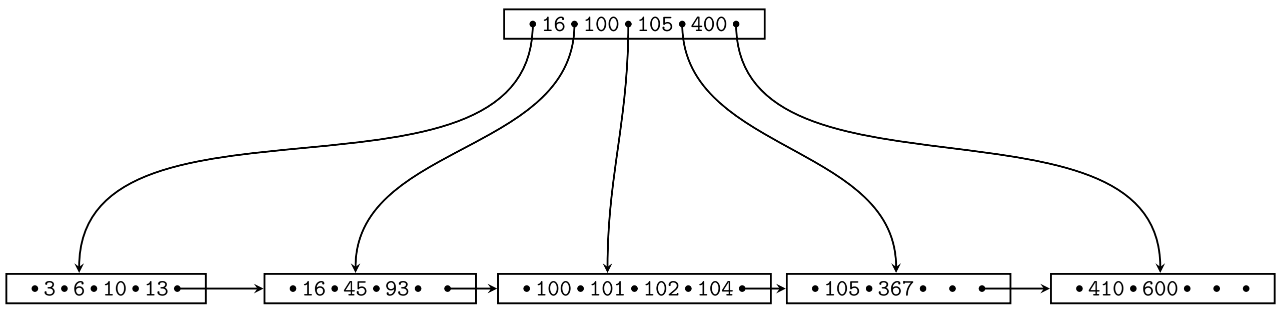


Question 1.3.2 Operations (10 Points)

Given is the B+-tree shown below ($n = 4$). Execute the following operations and write down the resulting B+-tree after each operation:

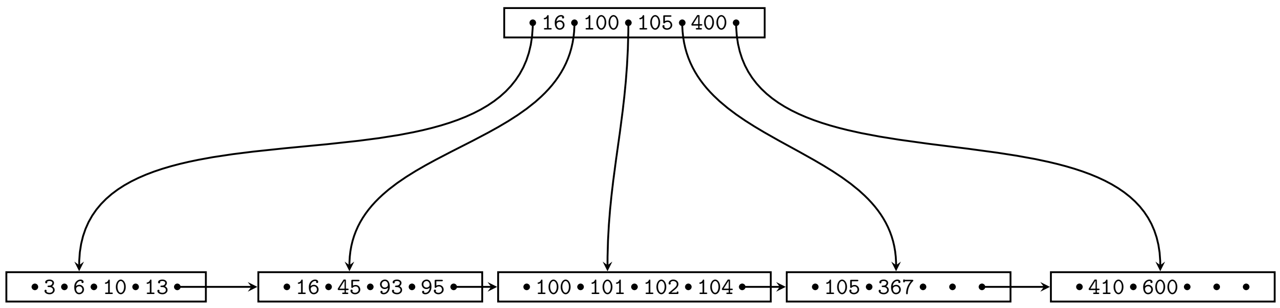
insert(95), insert(96), insert(103), delete(3), delete(6), delete(13)

Use the conventions for splitting and merging introduced in the previous question.

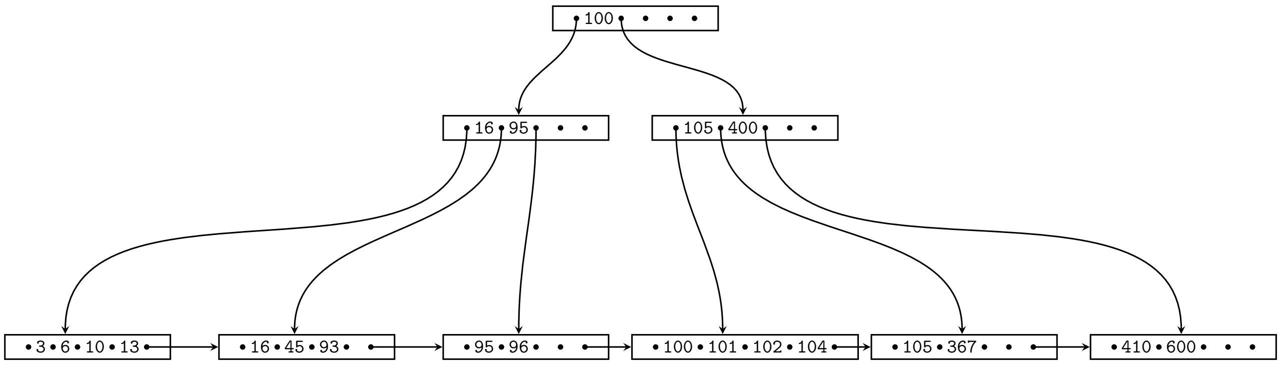


Solution

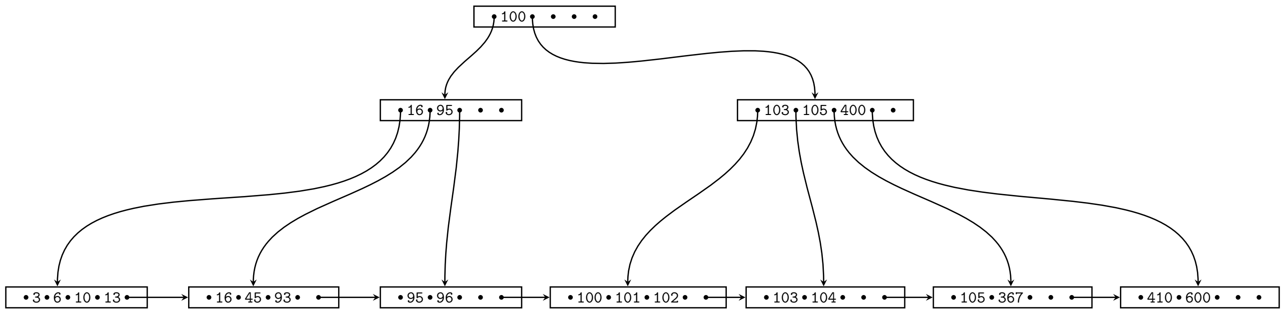
Insert 95



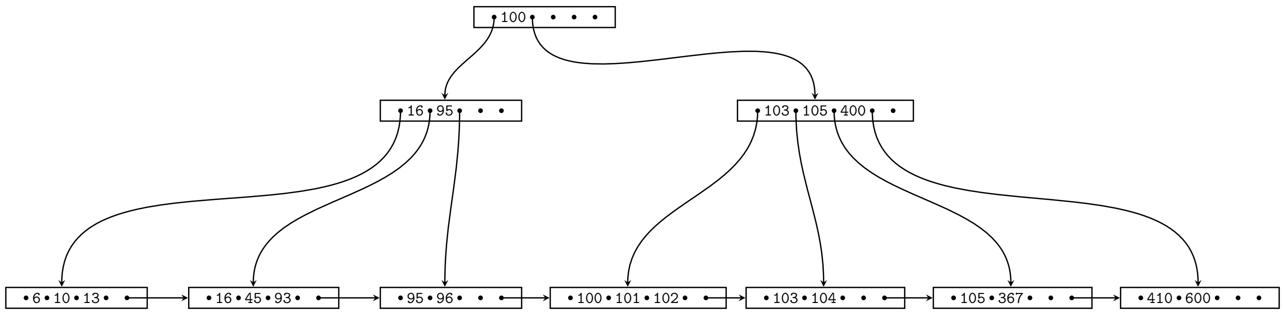
Insert 96



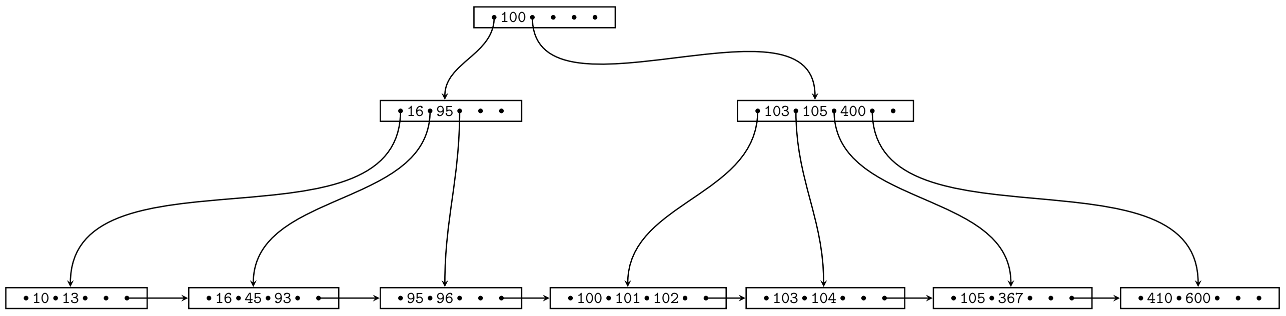
Insert 103



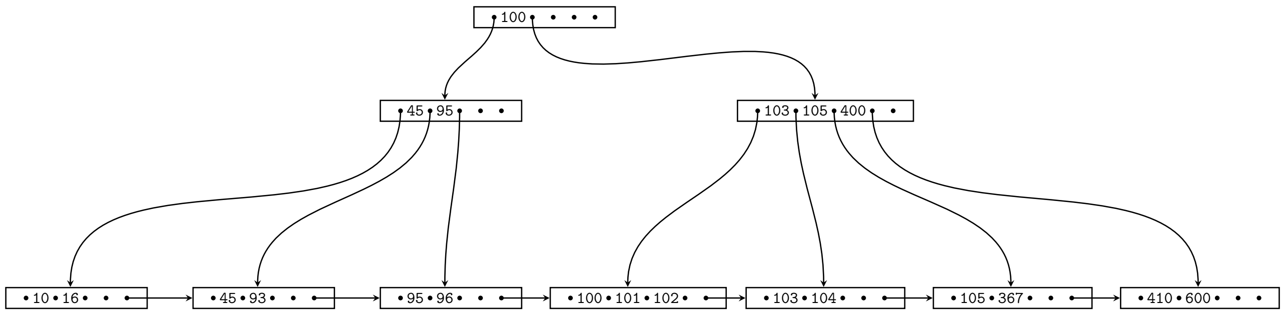
Delete 3



Delete 6



Delete 13



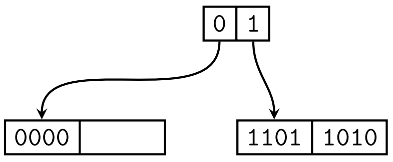
Question 1.3.3 Extensible Hashing (8 Points)

Consider the extensible Hash index shown below that is the result of inserting values 0, 1, and 2. Each page holds two keys. Execute the following operations

`insert(4),insert(1),insert(7),insert(7)`

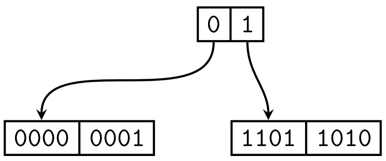
and write down the resulting index after each operation. Assume the hash function is defined as:

x	h(x)
0	1101
1	0000
2	1010
3	1100
4	0001
5	0000
6	1010
7	0111
8	1110

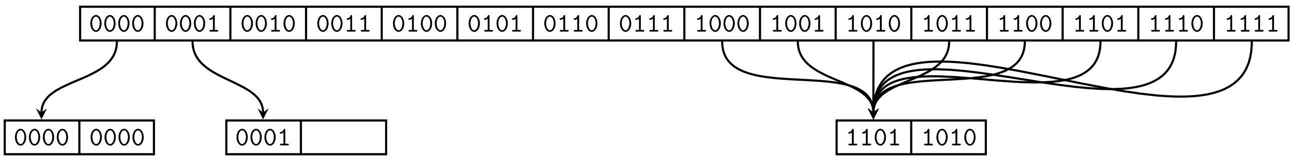


Solution

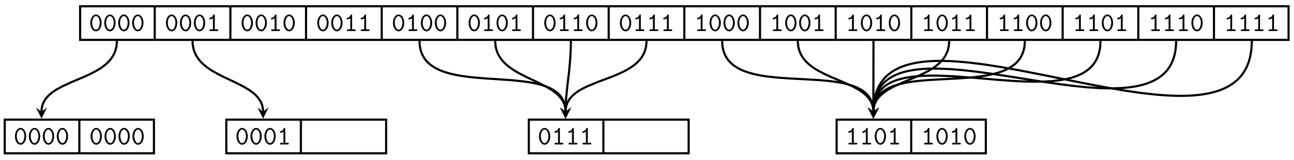
insert(4)



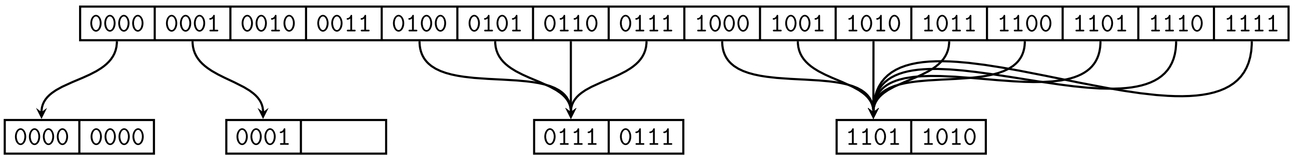
insert(1)



insert(7)



insert(7)



Part 1.4 Result Size Estimations (Total: 10 Points)

Consider a table `lecture` with attributes `title`, `campus`, `topic`, `roomSize`, a table `student` with `name`, `major`, `age`, and a table `attendsLecture` with attributes `student`, `lecture`, and `hoursAttended`. `attendsLecture.student` is a foreign key to `student`. Attribute `lecture` of relation `attendsService` is a foreign key to of relation `lecture`. Given are the following statistics:

$$\begin{aligned} T(lecture) &= 10 & T(student) &= 8,000 & T(attendsLecture) &= 40,000 \\ V(lecture, title) &= 10 & V(student, name) &= 7,500 & V(attendsLecture, student) &= 8,000 \\ V(lecture, campus) &= 2 & V(student, major) &= 4 & V(attendsLecture, lecture) &= 8 \\ V(lecture, topic) &= 3 & V(student, age) &= 40 & V(attendsLecture, hoursAttended) &= 100 \\ V(lecture, roomSize) &= 5 \end{aligned}$$

Question 1.4.1 Estimate Result Size (3 Points)

Estimate the number of result tuples for the query $q = \sigma_{topic=DB}(lecture)$ using the first assumption presented in class (values used in queries are uniformly distributed within the active domain).

Solution

$$T(q) = \frac{T(lecture)}{V(lecture, topic)} = \frac{10}{3} \simeq 3.3$$

Question 1.4.2 Estimate Result Size (3 Points)

Estimate the number of result tuples for the query $q = \sigma_{hoursAttended<20}(attendsLecture)$ using the first assumption presented in class. The minimum and maximum values of attribute *hoursAttended* are 0 and 100.

Solution

$$\begin{aligned}T(q) &= \frac{20-0}{max(hoursAttended) - min(hoursAttended) + 1} \times T(attendsLecture)\\ &= \frac{20}{101} \times 40,000 \simeq 7921\end{aligned}$$

Question 1.4.3 Estimate Result Size (4 Points)

Estimate the number of result tuples for the query q below using the first assumption presented in class.

$$q = (student \bowtie_{name=student} \sigma_{hoursAttended=15}(attendsLecture) \bowtie_{lecture=title} lecture$$

Solution

$$\begin{aligned}q_1 &= \sigma_{hoursAttended=15}(attendsLecture)\\ T(q_1) &= \frac{T(attendsLecture)}{V(attendsLecture, hoursAttended)} = \frac{40,000}{100} = 400\\ V(q_1, student) &= min(V(attendsLecture, student), T(q_1)) = 400\\ V(q_1, lecture) &= V(attendsLecture, lecture) = 8\end{aligned}$$

$$\begin{aligned}T(q) &= \frac{T(student) \times T(q_1) \times T(lecture)}{max(V(student, name), V(q_1, student)) \times max(V(q_1, lecture), V(lecture, title))}\\ &= \frac{8,000 \times 400 \times 10}{max(7,500, 400) \times max(8, 10)} \approx 426.66\end{aligned}$$