Åbo Akademi University

Advance Course on Database

# Weekly Report 3

Luis Araújo(2004624)

May 27, 2021

# Contents

# Chapter 1

# Introduction

In this weekly exercise, I was the chairman, and once the work flow was working, we decided to continue the same strategy of work. The topics approached were Hashing and bitmap indices.

# Chapter 2

# Exercises

**2.1 1) Assume that we have an extendable hash index on a file with an integer search key, and that the hash function is h(x) = x mod 8 (i.e. the hash value is in the range [0,7]) and a bucket can contain at most three records. Draw an illustration of how the extendable hash structure develops, step by step,when inserting records with the following search key values:**
**2, 3, 5, 7, 11, 17, 19, 23, 29, 31**

**Insert 2, 3 and 5**

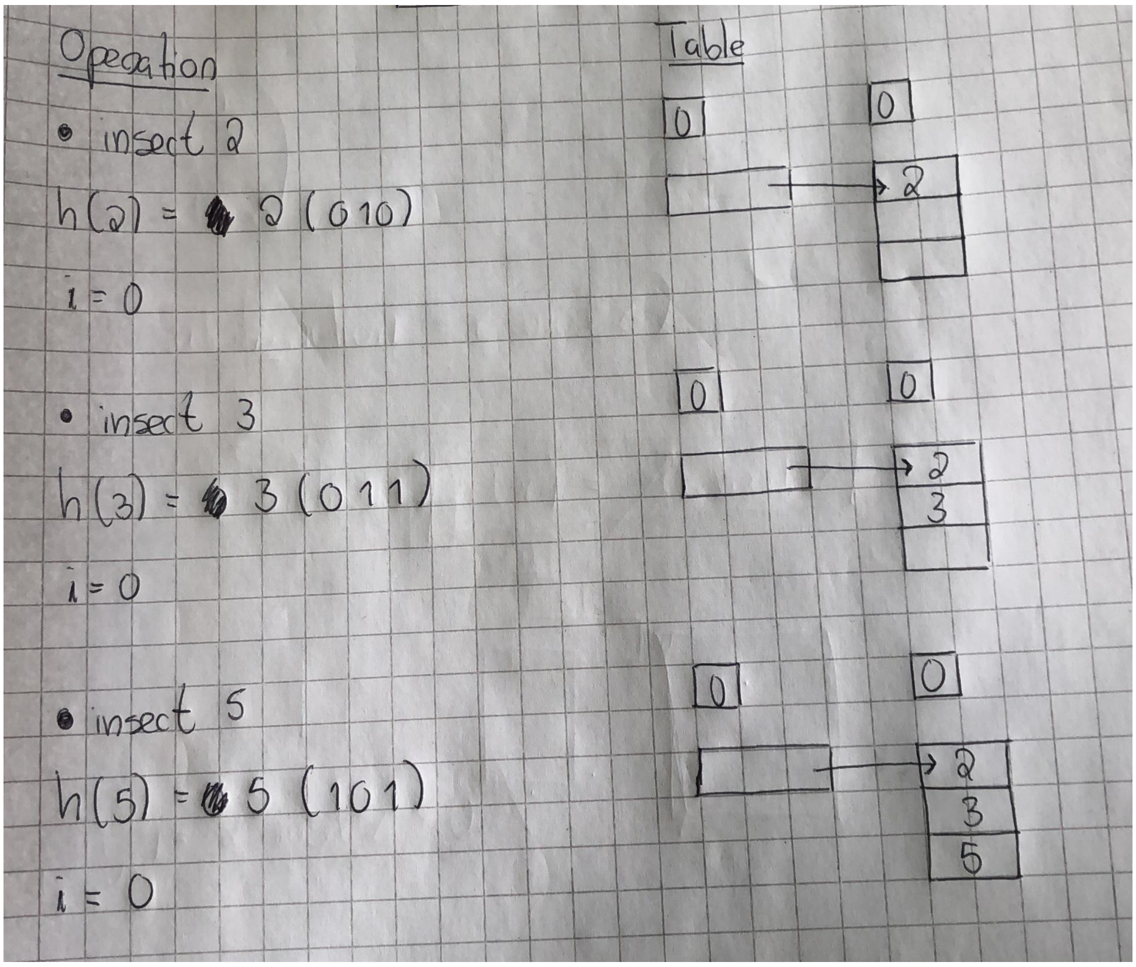


Figure 2.1: Insert 2, 3 and 5

**Insert 7, 11 and 17**

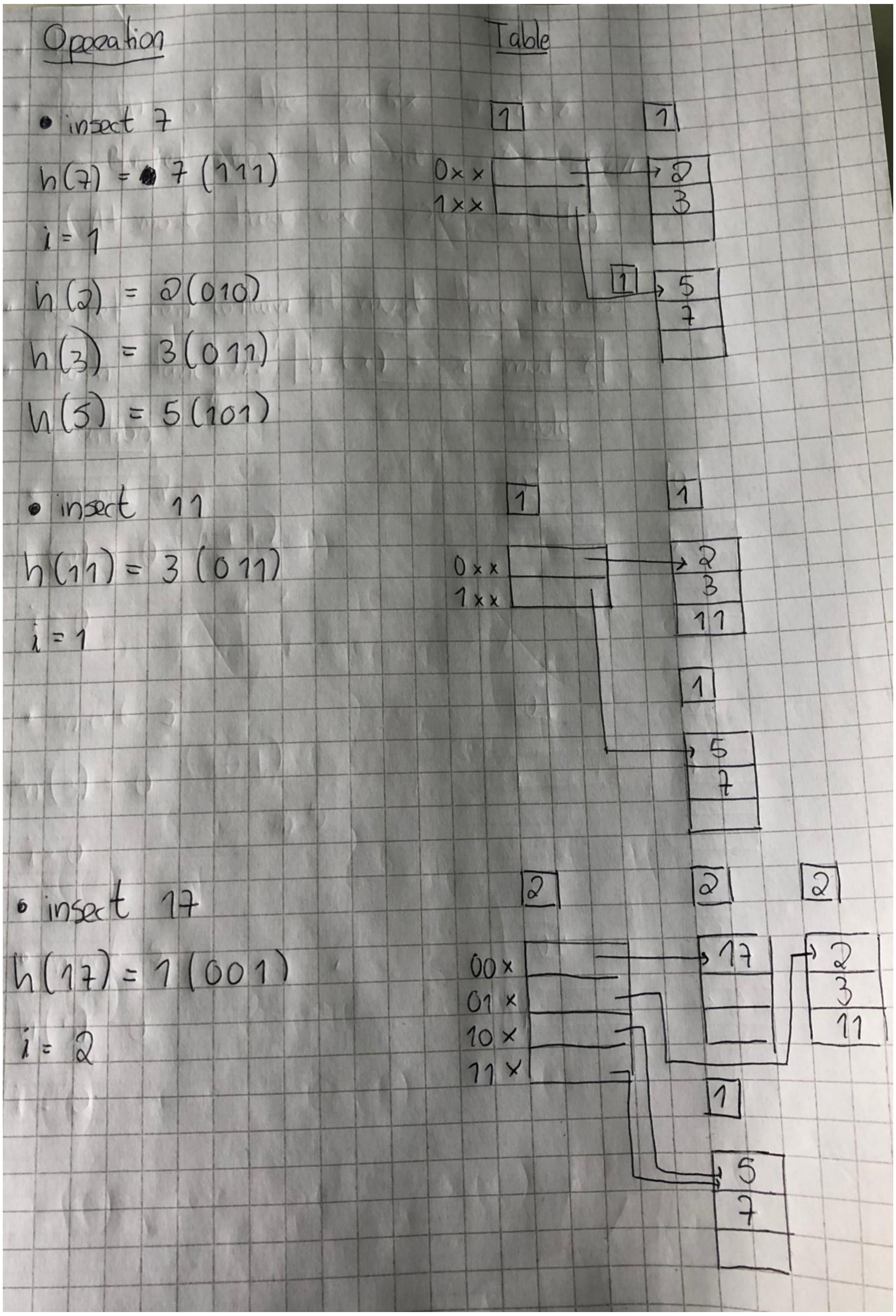


Figure 2.2: Insert 7, 11 and 17

**Insert 19, 23 and 29**

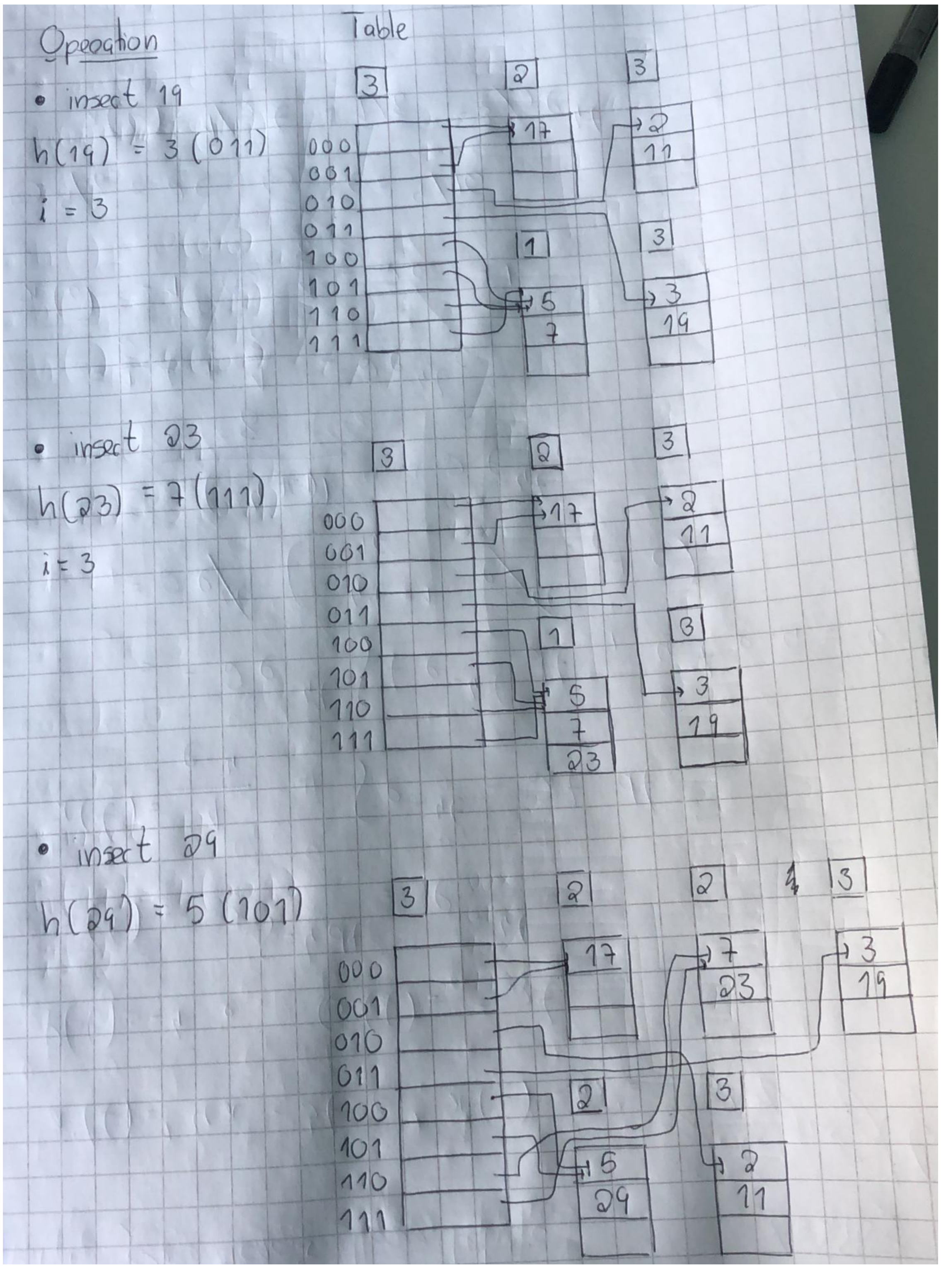


Figure 2.3: Insert 19, 23 and 29

**Insert 31**

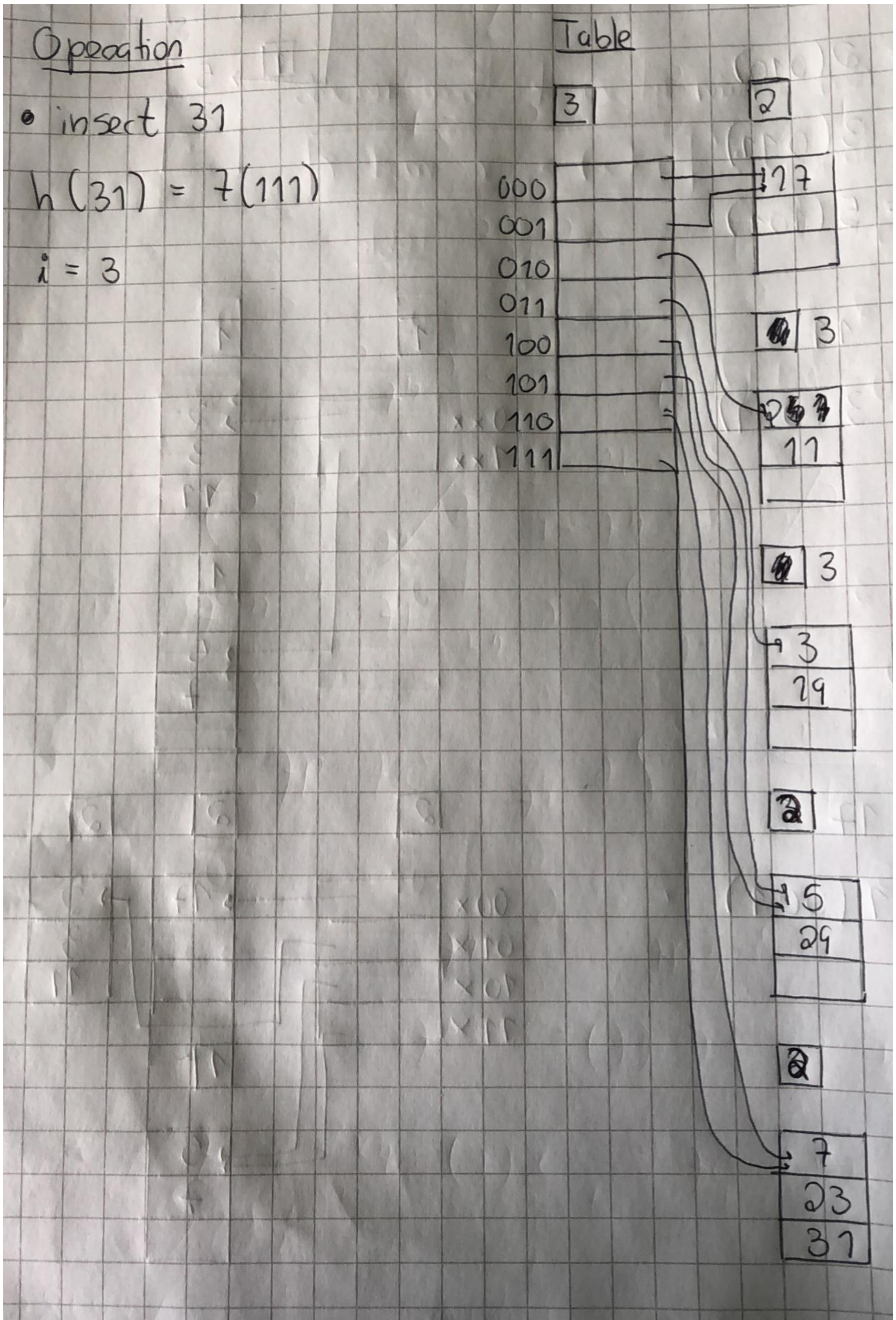


Figure 2.4: Insert 31

**Review**

This exercise is almost completely correct, the only mistake take by the team was not realising that when duplicating the bucket address table on the insertion of the value 19, instead of putting the value 11 on the index 011, by a lack of attention we putted on the index 010.

## 2.2 2) What will happen if we have a file with an extendable hash index and insert a very large number of records all with the same search key? As an example, we could have an index on the family name in a phone number catalog, and insert a very large number of records with the key "Andersson". How will this affect the performance of the index?

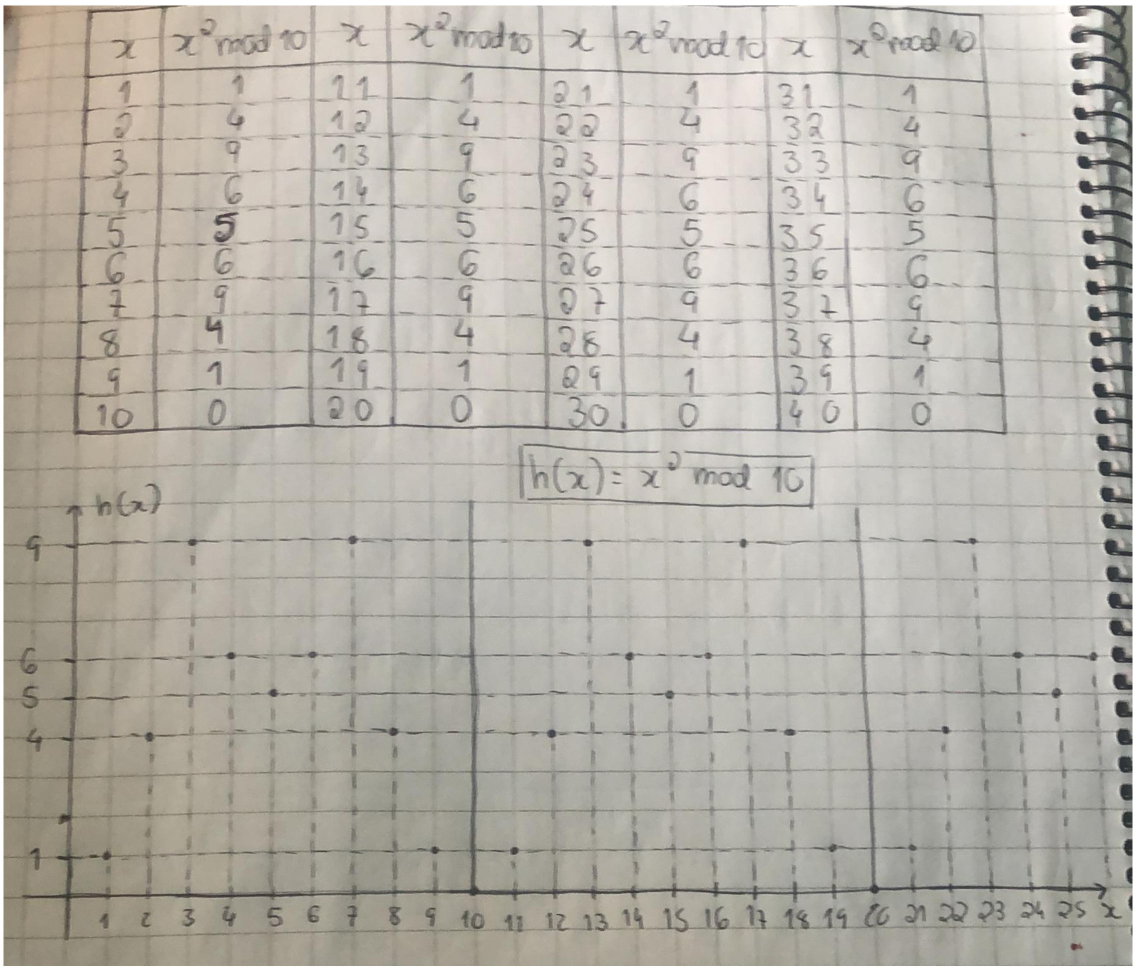
Performance of the index does not degrade with the growth of file because the structure grows dynamically on demand

**Review**

Unfortunately, the question was not well understood by the team. After, discussing and analysing the answer of the teacher we understood that the records would go to the same bucket and the bucket will be always full, creating an overflow of buckets.

## 2.3 3) Assume we have a hash function defined as $h(x) = x^2\ mod\ B$ where B is the number of buckets.

### 2.3.1 a) What is wrong with this hash function if B = 10?



| $x$ | $x^2$ mod 10 | $x$ | $x^2$ mod 10 | $x$ | $x^2$ mod 10 | $x$ | $x^2$ mod 10 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 11 | 1 | 21 | 1 | 31 | 1 |
| 2 | 4 | 12 | 4 | 22 | 4 | 32 | 4 |
| 3 | 9 | 13 | 9 | 23 | 9 | 33 | 9 |
| 4 | 6 | 14 | 6 | 24 | 6 | 34 | 6 |
| 5 | 5 | 15 | 5 | 25 | 5 | 35 | 5 |
| 6 | 6 | 16 | 6 | 26 | 6 | 36 | 6 |
| 7 | 9 | 17 | 9 | 27 | 9 | 37 | 9 |
| 8 | 4 | 18 | 4 | 28 | 4 | 38 | 4 |
| 9 | 1 | 19 | 1 | 29 | 1 | 39 | 1 |
| 10 | 0 | 20 | 0 | 30 | 0 | 40 | 0 |

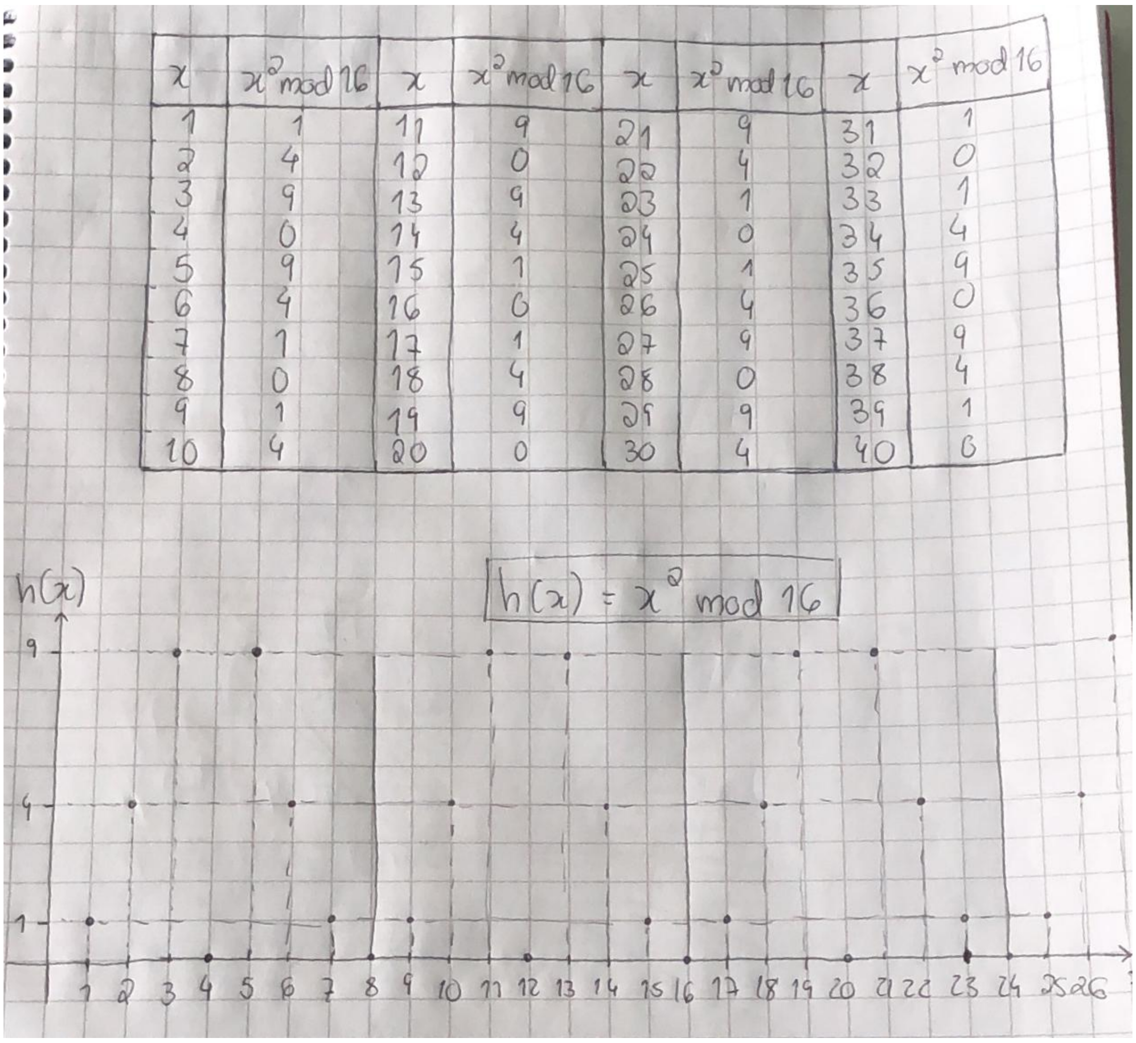
Figure 2.5

The problem with with this hash function if B = 10 is that it is a periodical function (of period 10), like we can see in Figure 5. Which this implies for the purpose of an hash table is that no matter what is the value x the result h(x) will always be between a set of results. In this particular case h(x) varies between 0, 1, 4, 5, 6 and 9, this results in the utilization of only this buckets and over time in a lot of overflow in these. For this reasons, this is a sub-optimal hash function.

### 2.3.2 Review

Correct exercise.

### 2.3.3 b) Is the hash function we get with B = 16 better or worse?



| $x$ | $x^2 \bmod 16$ | $x$ | $x^2 \bmod 16$ | $x$ | $x^2 \bmod 16$ | $x$ | $x^2 \bmod 16$ |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 11 | 9 | 21 | 9 | 31 | 1 |
| 2 | 4 | 12 | 0 | 22 | 4 | 32 | 0 |
| 3 | 9 | 13 | 9 | 23 | 1 | 33 | 1 |
| 4 | 0 | 14 | 4 | 24 | 0 | 34 | 4 |
| 5 | 9 | 15 | 1 | 25 | 1 | 35 | 9 |
| 6 | 4 | 16 | 0 | 26 | 4 | 36 | 0 |
| 7 | 1 | 17 | 1 | 27 | 9 | 37 | 9 |
| 8 | 0 | 18 | 4 | 28 | 0 | 38 | 4 |
| 9 | 1 | 19 | 9 | 29 | 9 | 39 | 1 |
| 10 | 4 | 20 | 0 | 30 | 4 | 40 | 0 |

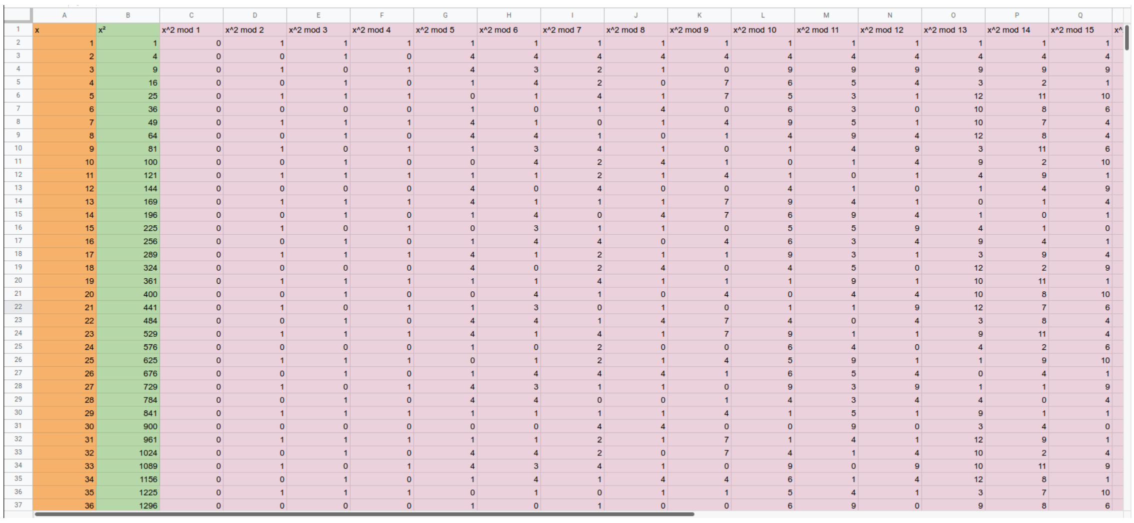
Figure 2.6

As we can see in Figure 6, the hash function we get with B = 16 is worse because is also a periodic function but with a smaller period - eight - and the results h(x) vary between a smaller set of values as a consequence: 0, 1, 4 and 9. The problems exposed above are even worse in this case.

### 2.3.4 Review

Correct exercise.

### 2.3.5 c) Are there any values of B that give a useful hash function?



| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | x | x² | x^2 mod 1 | x^2 mod 2 | x^2 mod 3 | x^2 mod 4 | x^2 mod 5 | x^2 mod 6 | x^2 mod 7 | x^2 mod 8 | x^2 mod 9 | x^2 mod 10 | x^2 mod 11 | x^2 mod 12 | x^2 mod 13 | x^2 mod 14 | x^2 mod 15 | x^ |
| 2 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| 3 | 2 | 4 | 0 | 0 | 1 | 0 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | |
| 4 | 3 | 9 | 0 | 1 | 0 | 1 | 4 | 3 | 2 | 1 | 0 | 9 | 9 | 9 | 9 | 9 | 9 | |
| 5 | 4 | 16 | 0 | 0 | 1 | 0 | 1 | 4 | 2 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | |
| 6 | 5 | 25 | 0 | 1 | 1 | 1 | 0 | 1 | 4 | 1 | 7 | 5 | 3 | 1 | 12 | 11 | 10 | |
| 7 | 6 | 36 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 4 | 0 | 6 | 3 | 0 | 10 | 8 | 6 | |
| 8 | 7 | 49 | 0 | 1 | 1 | 1 | 4 | 1 | 0 | 1 | 4 | 9 | 5 | 1 | 10 | 7 | 4 | |
| 9 | 8 | 64 | 0 | 0 | 1 | 0 | 4 | 4 | 1 | 0 | 1 | 4 | 9 | 4 | 12 | 8 | 4 | |
| 10 | 9 | 81 | 0 | 1 | 0 | 1 | 1 | 3 | 4 | 1 | 0 | 1 | 4 | 9 | 3 | 11 | 6 | |
| 11 | 10 | 100 | 0 | 0 | 1 | 0 | 0 | 4 | 2 | 4 | 1 | 0 | 1 | 4 | 9 | 2 | 10 | |
| 12 | 11 | 121 | 0 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 4 | 1 | 0 | 1 | 4 | 9 | 1 | |
| 13 | 12 | 144 | 0 | 0 | 0 | 0 | 4 | 0 | 4 | 0 | 0 | 4 | 1 | 0 | 1 | 4 | 9 | |
| 14 | 13 | 169 | 0 | 1 | 1 | 1 | 4 | 1 | 1 | 1 | 7 | 9 | 4 | 1 | 0 | 1 | 4 | |
| 15 | 14 | 196 | 0 | 0 | 1 | 0 | 1 | 4 | 0 | 4 | 7 | 6 | 9 | 4 | 1 | 0 | 1 | |
| 16 | 15 | 225 | 0 | 1 | 0 | 1 | 0 | 3 | 1 | 1 | 0 | 5 | 5 | 9 | 4 | 1 | 0 | |
| 17 | 16 | 256 | 0 | 0 | 1 | 0 | 1 | 4 | 4 | 0 | 4 | 6 | 3 | 4 | 9 | 4 | 1 | |
| 18 | 17 | 289 | 0 | 1 | 1 | 1 | 4 | 1 | 2 | 1 | 1 | 9 | 3 | 1 | 3 | 9 | 4 | |
| 19 | 18 | 324 | 0 | 0 | 0 | 0 | 4 | 0 | 2 | 4 | 0 | 4 | 5 | 0 | 12 | 2 | 9 | |
| 20 | 19 | 361 | 0 | 1 | 1 | 1 | 1 | 1 | 4 | 1 | 1 | 1 | 9 | 1 | 10 | 11 | 1 | |
| 21 | 20 | 400 | 0 | 0 | 1 | 0 | 0 | 4 | 1 | 0 | 4 | 0 | 4 | 4 | 10 | 8 | 10 | |
| 22 | 21 | 441 | 0 | 1 | 0 | 1 | 1 | 3 | 0 | 1 | 0 | 1 | 1 | 9 | 12 | 7 | 6 | |
| 23 | 22 | 484 | 0 | 0 | 1 | 0 | 4 | 4 | 1 | 4 | 7 | 4 | 0 | 4 | 3 | 8 | 4 | |
| 24 | 23 | 529 | 0 | 1 | 1 | 1 | 4 | 1 | 4 | 1 | 7 | 9 | 1 | 1 | 9 | 11 | 4 | |
| 25 | 24 | 576 | 0 | 0 | 0 | 0 | 1 | 0 | 2 | 0 | 0 | 6 | 4 | 0 | 4 | 2 | 6 | |
| 26 | 25 | 625 | 0 | 1 | 1 | 1 | 0 | 1 | 2 | 1 | 4 | 5 | 9 | 1 | 1 | 9 | 10 | |
| 27 | 26 | 676 | 0 | 0 | 1 | 0 | 1 | 4 | 4 | 4 | 1 | 6 | 5 | 4 | 0 | 4 | 1 | |
| 28 | 27 | 729 | 0 | 1 | 0 | 1 | 4 | 3 | 1 | 1 | 0 | 9 | 3 | 9 | 1 | 1 | 9 | |
| 29 | 28 | 784 | 0 | 0 | 1 | 0 | 4 | 4 | 0 | 0 | 1 | 4 | 3 | 4 | 4 | 0 | 4 | |
| 30 | 29 | 841 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 4 | 1 | 5 | 1 | 9 | 1 | 1 | |
| 31 | 30 | 900 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 4 | 0 | 0 | 9 | 0 | 3 | 4 | 0 | |
| 32 | 31 | 961 | 0 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 7 | 1 | 4 | 1 | 12 | 9 | 1 | |
| 33 | 32 | 1024 | 0 | 0 | 1 | 0 | 4 | 4 | 2 | 0 | 7 | 4 | 1 | 4 | 10 | 2 | 4 | |
| 34 | 33 | 1089 | 0 | 1 | 0 | 1 | 4 | 3 | 4 | 1 | 0 | 9 | 0 | 9 | 10 | 11 | 9 | |
| 35 | 34 | 1156 | 0 | 0 | 1 | 0 | 1 | 4 | 1 | 4 | 4 | 6 | 1 | 4 | 12 | 8 | 1 | |
| 36 | 35 | 1225 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 5 | 4 | 1 | 3 | 7 | 10 | |
| 37 | 36 | 1296 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 6 | 9 | 0 | 9 | 8 | 6 | |

Figure 2.7

For B = 1 (meaning 1 bucket), the hash function looks like this:

$h(x) = x^2 \bmod 1$

In this scenario, the hash function will have only on output: zero, here we have a hash function that output to all of the buckets available and at a equal rate so we can consider in this case a useful hash function.

For B = 2 (meaning 2 buckets), the hash function looks like this:

$h(x) = x^2 \bmod 2$

In this scenario, the hash function will have binary output: zero if $x^2$ is pair and one if it is impair, here we have a hash function that output to all of the buckets available and at a equal rate so we can consider in this case a useful hash function.

In any other scenario, it is not a useful hash function.

### 2.3.6 Review

Correct exercise.

## 2.4 4) Go through the tables in the university database and have a look at how the indices are created in the schema definitions (in the SQL file university.sql). Verify that the expected indices exist by using the command show index from table_name for each table.

### 2.4.1 a) Which tables have both a primary and a secondary index?

The tables that have both a primary and secondary index are:

- advisor
- course
- instructor
- prereq
- section
- student
- takes
- teaches

### 2.4.2 Review

Correct exercise.

### 2.4.3 b) Which tables have a primary index on a composite key (i.e. a primary key that consists of more than one attribute)?

The tables that have a primary index on a composite key are:

- classroom
- prereq
- section
- takes
- teaches
- time_slot

### 2.4.4 Review

Correct exercise.

### 2.4.5 c) Which tables have a secondary index on a composite key?

The tables that have a secondary index on a composite key are:

- section
- takes
- teaches

### 2.4.6 Review

Correct exercise.

5)

### 2.4.7 a) Construct bitmap indices on the attributes dept_name and salary. The salary values should be divided into 4 ranges: 0–49999, 50000–59999, 60000–69999 and 70000 and above

Constructing the bitmap for the different departments, first selected them, and after we attributed the corresponded bit, 1 if the record was present in that position and 0 if it was not present.

| Bitmaps for dept_name | | |
|---|---|---|
| # | Dept_name | Bitmap |
| 1 | Comp.Sci. | 100000100010 |
| 2 | Finance | 010000001000 |
| 3 | Music | 001000000000 |
| 4 | Physics | 000101000000 |
| 5 | History | 000010010000 |
| 6 | Biology | 000000000100 |
| 7 | Elec. Eng. | 000000000001 |

For the salary level bitmap, we first created a table were in each row we could distinguish the salary and the level corresponded.
After this first table been created, the final table was effortlessly created.

| Record number | Salary (level) |
|---|---|
| 0 | 65000 (L3) |
| 1 | 90000 (L4) |
| 2 | 40000 (L1) |
| 3 | 95000 (L4) |
| 4 | 60000 (L3) |
| 5 | 87000 (L4) |
| 6 | 75000 (L4) |
| 7 | 62000 (L3) |
| 8 | 80000 (L4) |
| 9 | 72000 (L4) |
| 10 | 92000 (L4) |
| 11 | 80000 (L4) |

| Bitmaps for salary_level | |
|---|---|
| Level | Bitmap |
| L1 | 001000000000 |
| L2 | 000000000000 |
| L3 | 100010010000 |
| L4 | 010101101111 |

### 2.4.8 Review

Correct exercise.

**b) Consider a SQL query that requests all instructors in the Finance department:**

```
SELECT ID, name, salary
FROM instructor
WHERE dept_name = 'Finance' AND salary >= 80000;
```

**Describe how this query could be answered using the above created bitmap indexes. Describe the intermediate steps and bit operations in the execution of the query.**

To answer this query, using the bitmap indexes created, first we have to compute:
$dept\_name = 'Finance'\ and\ salary\ >=\ 80000$.

We can easily notice that the bitmap from the department 'Finance' is 010000001000.
One the other hand, it's not that simple to obtain the other part of the operation, $salary\ >=$ 80000. Initially, we can identify that this operation is included on the level 4 of the salary_level bitmap, but, and since this level is determined by all the salaries higher than 70000, this level has also the salary between 7000 and 8000, which we need to be a little careful when getting the determined bitmap. So, with this knowledge, we can get the bitmap from the salary_level 4, 010101101111, and verify each position where the bit is 1. If the salary is lower than 80000 we change the bit to 0, otherwise the bit stays as 1. With this, we obtain the bitmap 010101001011.Then, we can determine the final intersection operation with this 2 bitmaps:
010000001000 $AND$ 010101001011 $=$ 010000001000

Finally, with this bitmap we can obtain the positions of the records needed to solve this query, which are positions number 1 and 8 of the table.

### 2.4.9 Review

Correct exercise.

# Chapter 3

# Conclusion

It's noticeable from the correct exercises that the group had some knowledge in hashing and a large understanding of bitmaps. Even though, this comprehension the exercises were a good way of solidifying the better understanding of the topic.