Consider the following relational schema of a database that stores clinical records of the IST Lisbon Hospital:

Health-professional (professional-id, name, h-birth-date, role, salary, email, phone, performance-evaluation, start-date)
unique(email)
not null(email)

Stock (stock-id, type, service, description, quantity, delivered-date, need-to-order)

Schedule (schedule-id, professional-id, slot-start, slot-end)
professional-id: FK(Health-professional)

Patient (patient-id, name, p-birth-date)

Occurrence (patient-id, occurrence-id, type, occurrence-date, hospitalization)
patient_id: FK(Patient)

and the following tuples as examples:

Health-professional
(1, 'Catarina', 20-09-1995, ´nurse´, ´substitution´, 1205, catarina@chlc.pt, 919999999, 20, 19-01-2021)
Stock
(0, ´needle´, ´ER´, ´2ml´ , 39 , 16-04-2021, True)
Schedule
(1, 2021-06-01 00:00:00, 2021-06-01 08:00:01)
Patient
(37, ´Gervásio Lopes´, 01-01-1934)
Occurrence
(37, 40, ´Hypertension´, 19-04-2021 09:30:00, False)

Create the database in SQL Server and insert some tuples in the tables (you may want to insert the above tuples and some others)

Question 1 – SQL Server Storage and Indexing

a) Write the T-SQL command to create a database with the characteristics that follow. The database should be named IST-LH-Records. Consider the Covid-19 scenario for file growth, where at any point the number of patient records can increase exponentially, and the db

should follow this growth pattern. The DB should have two data files in different filegroups, each of them with an initial size of 23 MB. It must also have a log file with an initial size of 13 MB. The maximum size available on all the disks dedicated to the DB is 5 TB.

Answer:

```
CREATE DATABASE  IST_LH_Records ON
 PRIMARY (
            NAME = 'IST_LH_Record_1',
            FILENAME = 'C:\Data\Primary\LH_RecordsDB_1.mdf',
            SIZE = 23MB,
            FILEGROWTH = 100%,
            MAXSIZE = 2TB
),

FILEGROUP 'IST_LH_Record_2' (
            NAME = LH_RecordsDB_2,
            FILENAME = 'C:\Data\Secondary\LH_RecordsDB_2.ndf',
            SIZE = 23MB,
            FILEGROWTH = 100%,
            MAXSIZE = 2TB
            )

LOG ON (
            NAME = 'IST_LH_Record_Log',
            FILENAME = 'C:\Data\Log\LH_RecordsDB_log.ldf',
            SIZE = 13MB,
            FILEGROWTH = 100%,
            MAXSIZE =1TB
            );
```

b) Present the T-SQL instructions to create all tables. The Occurrence table is the result of a data migration that took place on 30-04-2021 at 3 PM. This table should be partitioned so that all tuples with occurrence-date more recent than the migration date are physically stored in the secondary filegroup, and all the other tuples are physically stored in the primary filegroup.

Answer:

```
CREATE TABLE Health_professional (
        professional_id INT PRIMARY KEY ,
        name VARCHAR(128),
        h_birth_date DATE,
        role VARCHAR(64),
        salary DECIMAL(9, 2),
        email VARCHAR(128) NOT NULL UNIQUE,
        phone VARCHAR(9),
        performance_evaluation DECIMAL(3, 1),
        start_date DATE,
        );

CREATE TABLE Stock(
        stock_id INT PRIMARY KEY,
        type VARCHAR(64),
        service VARCHAR(16),
        description VARCHAR(255),
        quantity INT,
        delivered_date DATE,
        need_to_order BIT,
);

CREATE TABLE Schedule(
        schedule_id INT PRIMARY KEY,
        professional_id INT FOREIGN KEY REFERENCES Health_professional(professional_id),
        slot_start DATETIME,
        slot_end DATETIME,
);

CREATE TABLE Patient(
        patient_id INT  PRIMARY KEY,
        name VARCHAR(128),
        p_birth_date DATE,
);


CREATE TABLE Occurrence(
        patient_id INT FOREIGN KEY REFERENCES Patient (patient_id),
        occurrence_id INT,
        type VARCHAR(64),
        occurrence_date DATETIME,
        hospitalization BIT,
        PRIMARY KEY (patient_id, occurrence_id),

);



CREATE PARTITION FUNCTION Occurrence_PartitionRange (DATETIME) AS RANGE LEFT FOR VALUES ('2021-04-30 15:00:00');

CREATE PARTITION SCHEME Occurrence_PartitionScheme AS PARTITION Occurrence_PartitionRange TO ([PRIMARY], IST_LH_Record_2);
```

c) Create T-SQL statements for creating one or more indexes to optimize the workload composed of the two following queries: i) get the ids of health professionals with a specific role (e.g., nurse), and ii) get the ids of health professionals older than a certain age. Explain your decision concerning the index types chosen.

Answer:

i)

To solve this you could use a hash non-clustered index over role and professional_id (covering). Hash since the search criteria is an equality. The implementation of such an index cannot be done in SQL Server. This is the case because of the native implementation of indexes in SQL Server, which is based in B+Trees.

```
CREATE NONCLUSTERED INDEX Role_index ON Health_professional(role, professional_id);
```

We also accepted:

Using an extra column on the Health_professional table to contain the hash of the role attribute. The index is created over that column.

```
ALTER TABLE Health_professional ADD roleHash AS checksum(role);

CREATE NONCLUSTERED INDEX IX_Health_professional_Role ON Health_professional(roleHash);
```

ii)

To optimize such a query you could use a b+tree non-clustered index on h_birth_date and professional_id, this is a covering index in this query.

```
CREATE NONCLUSTERED INDEX Index_Health_professional_BirthDate ON  Health_professional(h_birth_date,
professional_id);
```

Question 2 – B+ Trees

Consider that it is usual for people to ask for information about a patient based on his/her name. A B+ tree is built over the name attribute of the Patient table, holding 2 values of the search key per node. The following search key values are inserted, in the given order:

Arlindo Marques , José Oliveira, Joaquim Matos, David Jorge, Nuno Gonçalves, Daniel Mamede

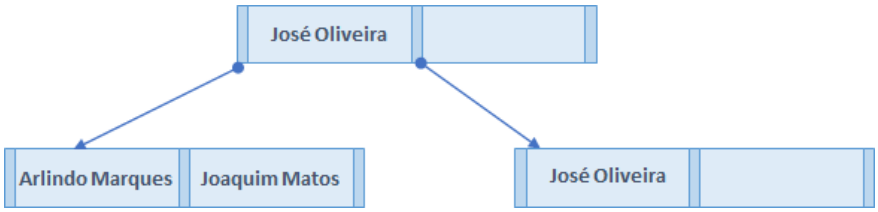
a) Draw the tree after each insertion.

Answer:

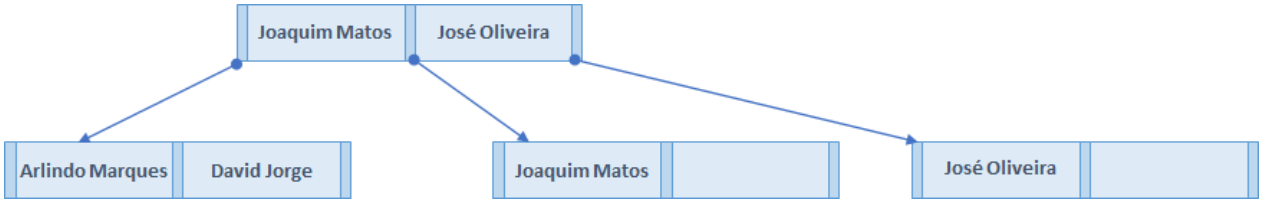
Insertion *Arlindo Marques*

Insertion *José Oliveira*

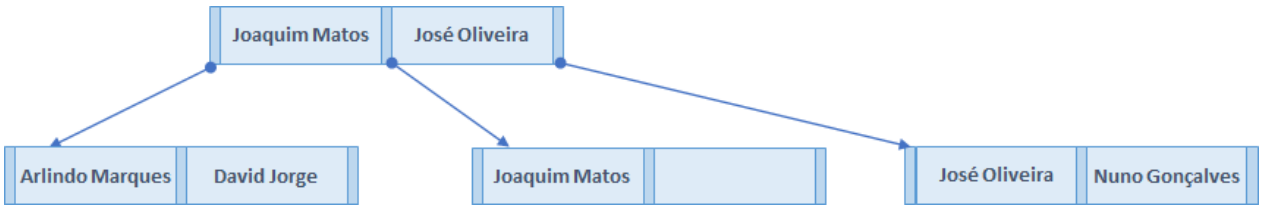
Insertion *Joaquim Matos*



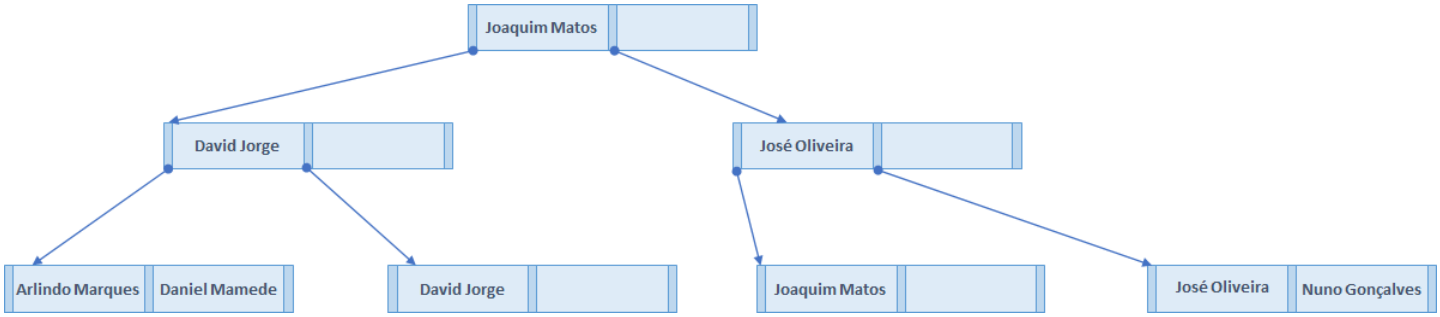
Insertion *David Jorge*



Insertion *Nuno Gonçalves*



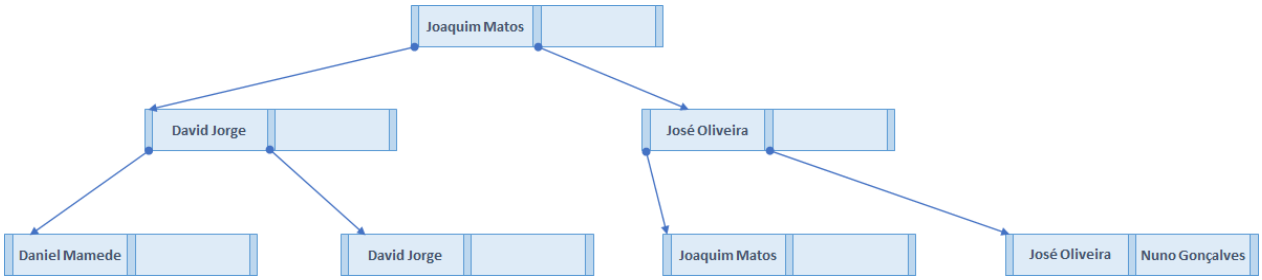
Insertion *Daniel Mamede*



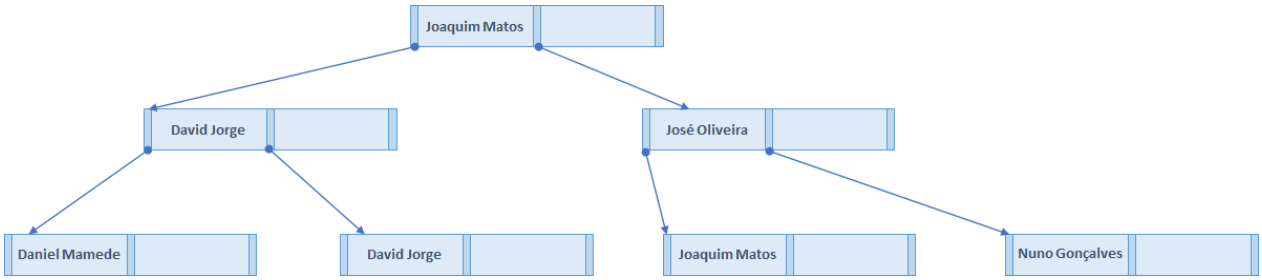
b) Using the final B+ tree resulting from the previous exercise, delete the following search keys, in the given order: *Arlindo Marques, José Oliveira, Joaquim Matos, David Jorge, Nuno Gonçalves, Daniel Mamede.* Present the tree after each deletion.

Answer:

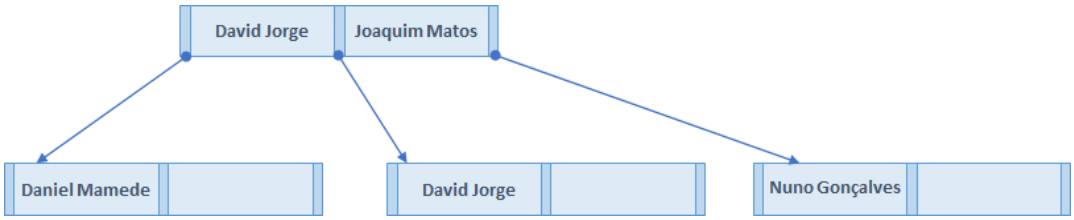
Deletion of *Arlindo Marques*



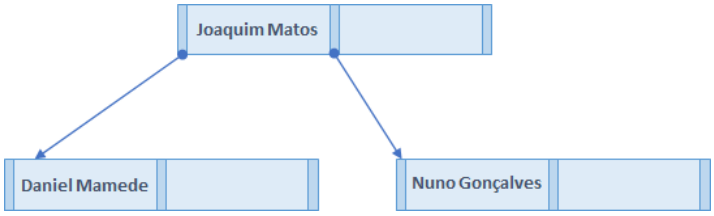
Deletion of *José Oliveira*



Deletion of *Joaquim Matos*



Deletion of *David Jorge*



Deletion of *Nuno Gonçalves*

Deletion of *Daniel Mamede*

Question 3 – Extendable Hashing

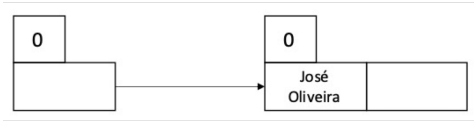
Consider that an extendable hashing index has been created over the name attribute of the Patient table where each bucket can hold up to 2 records. Show how the following records (with the given hash values) can be stored into such index, in the given order, **considering the most significant bits first**:

José Oliveira	`10001`
Arlindo Marques	`01000`
Joaquim Matos	`00100`
David Jorge	`10001`
Nuno Gonçalves	`10010`
Daniel Mamede	`01001`

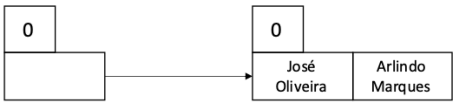
Draw the index after each insertion.

Answer:

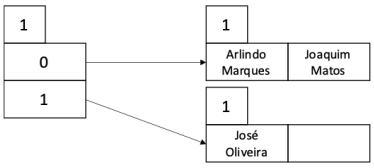
Insert José Oliveira



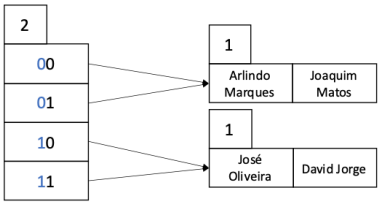
Insert Arlindo Marques



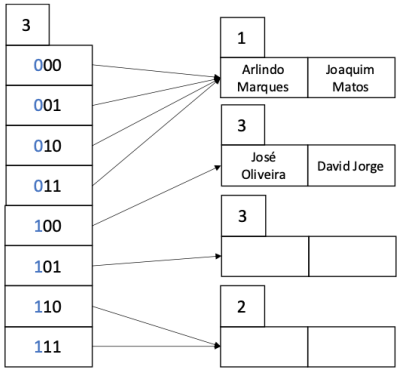
Insert Joaquim Matos

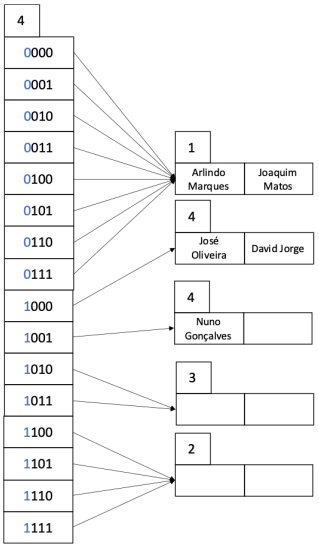


Insert David Jorge

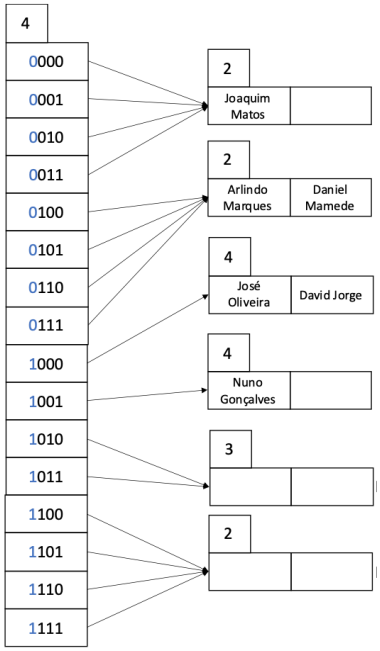


Insert Nuno Gonçalves





Insert Daniel Mamede



Question 4 – Query Processing and Optimization

a) Consider the Health-Professional table and a clustered B+tree over its salary attribute. Assume that this table has 729 tuples, and there are 27 records per block. Concerning the B+tree, consider that the number of pointers per node is 6. Estimate the cost of a selection (i.e., number of pages accessed), using the B+ tree index, that returns the tuples corresponding to the highest-earning professionals, that account for 1312 records.

Answer:

TreeHeight = $\lceil log_{\lceil n/2 \rceil}(K) \rceil = \lceil log_{\lceil 6/2 \rceil}(729) \rceil = 6$

b = Number of blocks containing matching records = 132/ 27 =5
Therefore we will need to read 5 blocks

Number of pages accessed = TreeHeigh + b = 6 + 5 = 11

b) Suppose you are joining the Health-Professional and Schedule tables. The Health-Professional table contains 3921 pages and the Schedule table contains 24601 pages. Each page of Health-Professional can accommodate 55 tuples and each page of Schedule can

accommodate 33 tuples. Compute the cost in terms of the number of I/Os (i.e., number of pages) of the following join algorithms:

1. Indexed nested loop join, using Schedule as the inner relation and using a B+tree over professional-id on Schedule. The index tree has a height of 7.

Answer:

b_H= number of blocks containing records of Health-Professional

n_H = number of tuples in the Health-Professional table

c = cost of a single selection on Schedule using the join condition = height of tree +1

R: $b_H + n_H$ * c = 3921 + 215655*8 = 1729161

2. Hash-join assuming that you have pages of memory and a fudge factor of 1.7. Indicate the best probe relation.

Answer:

$$number\ of\ partitions\ =\ \lceil \frac{b_H}{M} \rceil \text{ x f} =\ 1396 > 30$$

$$number\ of\ partitions\ =\ \lceil \frac{b_S}{M} \rceil \text{ x f} =\ 223 > 30$$

Which means we have to use recursive partitioning

$$2\left(b_S + b_H\right)\lceil log_{M-1}\left(b_S\right) - 1 \rceil + \left(b_S + b_H\right) =$$
$$2\left(3921 + 24601\right)\lceil log_{30-1}(3921) - 1 \rceil + \left(3921 + 24601\right) =$$
$$142610$$

$$2\left(b_H + b_S\right)\lceil log_{M-1}\left(b_H\right) - 1 \rceil + \left(b_H + b_S\right) =$$
$$2\left(24601 + 3921\right)\lceil log_{30-1}(24601) - 1 \rceil + \left(24601 + 3921\right) =$$
$$199654$$

Using Health-Professional as build-in relation.

c) The first-ever child diagnosed with a new type of diabetes will be transferred to the hospital tomorrow. However, before her record is transferred, the hospital wants to check the name of the patients that have diabetes, were hospitalized in the past and were born after 2003. Write the query to obtain that information in relational algebra notation and estimate the number of tuples that result from the evaluation of the corresponding algebra expression. Consider that:

1. The number of tuples in Patient is 10.000 and in Occurrence is 30.000.

2. Hospitalization = {True, False} and a third of patients were hospitalized.
3. People with records in the hospital were born between 1921 and 2021.
4. There are 250 different types of occurrences.

Answer: :

$$\Pi_{name}(\sigma_{type='Diabetes' \wedge hospitalization='True'}(Occurrence) \bowtie \sigma_{p\text{-}birth\text{-}date >= 2004}(Patient))$$

nO = number of tuples in Occurrence
nP= number of tuples in Patient

Number of tuples resulting from:

$$\sigma_{type='Diabetes' \wedge hospitalization='True'}(Occurrence)$$

$$nO \times \frac{s1 * s2}{nO^{2}} = 30000 \times \frac{10000 * 120}{30000^{2}} = 40$$

Number of tuples resulting from:

$$\sigma_{p\text{-}birth\text{-}date >= 2004}(Patient)$$

$$nP \times \frac{v - max(p{-}birth{-}date, Patient)}{min(p{-}birth{-}date, Patient) - max(p{-}birth{-}date, Patient)} =$$

$$10000 \times \frac{2004 - 2021}{1921 - 2021} = 1700$$

Numbers resulting from the join:

Patient ∩ Occurrence = patient-id

patient-id is key of Patient which means that each tuple of Occurence will join exactly one tuple of Patient therefore the number of tuples will be no greater than the number of tuples resulting from the selection over Occurrence (40). (will join exactly one tuple because is foreign key)

$$\Pi_{name}\ (\sigma_{type='Diabetes' \wedge hospitalization='True'}\ (Occurrence) \bowtie \sigma_{p\text{-}birth\text{-}date >= 2004}\ (Patient))$$

Assuming the worst case scenario where all names are different the number of tuples resulting from this expression will be 40

Question 5 – Transactions, Concurrency Control and Recovery Management

a) Consider two transactions T1 and T2 executing operations over tables Health-professional, and Patient. Give an example of a schedule that is possible under timestamp-based protocol (with Thomas Write rule) but not possible under two-phase locking and another example that is possible under two-phase locking but not possible under timestamp-based protocol (also with Thomas Write rule).

Answer:

T1	T2
Lock-S(H) R(H)	
	Lock-X(H) W(H)
W(P)	
W(H)	

T1	T2
R(H) **R-TS(H) = 1**	
	W(H) **W-TS(H) = 2**
W(P)**W-TS(H) = 1**	
W(H) **TS(T1) = 1 < W-TS(H) =2** **Thomas Rule ignore this write**	

It is possible under the timestamp-based protocol (with Thomas Rule), but not under the two-phase locking

T1	T2
Lock-S(H) R(H)	
	Lock-X(P) W(P) **Unlock(P)**
Lock-S(P) R(P)	

T1	T2
R(H) **R-TS(H) = 1**	
	W(P) **W-TS(H) = 2**
R(P) **TS(T1) = 1 < W-TS(H) = 2** **T1 is trying to read a value that was already overwritten**	

Possible under the two phase locking, but not under the timestamp-based protocol

b) Now, consider two transactions T3 and T4, where T3 writes two tuples of table Health-professional, t1(H) and t2(H), and T4 reads the same two tuples. Give an example where the timestamp test for a write operation fails and causes T3 to be restarted, in turn causing a cascade rollback of T4.

Answer: :

Consider TS(T3) <TS(T4)

T3	T4
W(t1(H))	
	R(t1(H))

	R(t2(H))
W(t2(H))	

c) Consider the following simplified representation for the log file, and suppose that the ARIES algorithm is executed by the recovery system:

LSN	Type	Transaction	Page
00	Begin_checkpoint	-	-
10	End_checkpoint	-	-
20	Update	T1	P1 (Patient)
30	Update	T2	P2 (Patient)
40	Commit	T1	
50	Update	T3	P2(Patient)
60	End	T1	
70	Update	T2	P3 (Patient)
80	Abort	T2	
	CRASH!!!		

Consider also that the system crashes during the recovery process from the crash that is represented above, after having persistently written two records in the log. Assume that the active transaction table and the dirty page table are empty at the time of the checkpointing. Show the contents of the log, the active transaction table, and the dirty page table after: (a) the analysis phase, (b) the redo phase, and (c) the undo phase that are executed during the whole recovery process.

Answer:

Analysis

ATT

T	Last LSN
T2	80
T3	50

DPT

Page	Rec LSN
P1	20
P2	30
P3	70

Redo

20	
30	Redo P2
40	
50	Redo P2
60	
70	Redo P3
80	

Undo

ToUndo = [80, 50]

80	ToUndo = [70,50]
70	Undo P3, write CLR, ToUndo = [50, 30]
50	Undo P2, write CLR, write END, ToUndo = [30]
30	Undo P2, write CLR, write END

Log

LSN	Prev LSN	XID	Type	PageID	UndoNext
90	80	T2	CLR	P3	30
100	50	T3	CLR	P2	null

110	100	T3	END	-	-
120	90	T2	CLR	P2	null
130	120	T2	END	-	-

d) At the IST Lisbon Hospital, there is a schedule manager that allocates the time slots for each health professional throughout the week. However, health professionals are rebelling for working too many hours per shift without payment. So, they need to be able to register their working hours in the corresponding relation (Schedule). You agreed to help them. Nurses are all registering their hours at the same time. To manage the rebellion, the manager is actively reading the Schedule relation.
    1. Assume that each nurse transaction is running with the read uncommitted isolation level. Discuss two possible anomalies that may occur while the manager is reading the Schedule and justify.
    2. Nurse Catarina decided to help her friend and record her time. However, her friend was also registering her own time simultaneously. Show through a pair of T-SQL transactions how a phantom read can occur.

Answer: :

1. non-repeatable reads or phantom reads can both occur in an uncommitted isolation level. A non-repeatable read could occur when a nurse is updating a row and the manager reading that row twice yielding different results. A phantom read could occur if the manager is retrieving the schedules from a day twice and the number of rows retrieved varies in that same query. This could happen if nurses are inserting rows in between those two queries in the ame transaction.
2.

Transaction of Nurse Catarina	Transaction of Nurse Duarte (Catarina's friend)
	SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED; BEGIN TRANSACTION; SELECT COUNT(*) FROM Schedule WHERE professionalId = 2;
BEGIN TRANSACTION; INSERT INTO Schedule	

(scheduleId,professionalId,slotStart,slotEnd) VALUES(1,1,2021-04-04 01:00:00,2021-04-04 05:00:00); COMMIT TRANSACTION;	
	SELECT COUNT(*) FROM Schedule WHERE professionalId = 2; **[returns a different count]** INSERT INTO Schedule (scheduleId,professionalId,slotStart,slotEnd) VALUES(1,1,2021-04-04 01:00:00,2021-04-04 05:00:00); COMMIT TRANSACTION;

Question 6 – Database Tuning

a) Consider the Schedule and Health-professional tables. Consider that the following queries are very frequent:
   - get the names of Health Professionals
   - get the nurses' schedule

Which indexes would you create over the two tables in order to improve both queries? Justify. Perform an experimental evaluation to check if the queries use the indexes you suggested and present the corresponding execution plans.

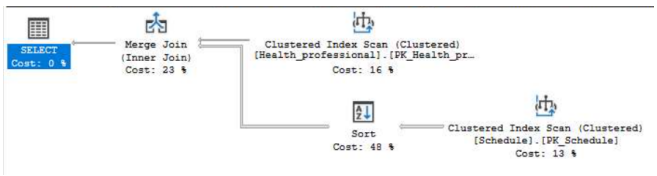
Answer:

There is no need for an index to get the names of Health Professionals, it will require a full table scan.

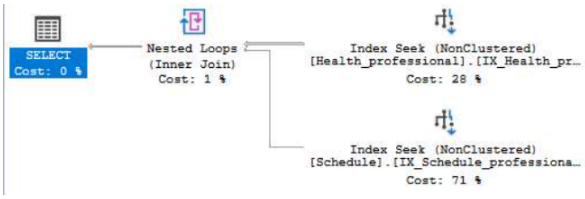
To get the nurses' schedule one can create a hash index over role in the Health professional table.
A non-clustered index on professional_id on the table Schedule to improve the performance of the join between Health_professional and Schedule.

```
ALTER TABLE Health_professional ADD roleHash AS checksum(role);

CREATE NONCLUSTERED INDEX IX_Health_professional_Role ON
        Health_professional(roleHash) INCLUDE (professional_id);

CREATE NONCLUSTERED INDEX IX_Schedule_professional_id ON
        Schedule(professional_id) INCLUDE (slot_start, slot_end);
```

Before an index scan followed by a sort is used.



After the index seek over the created index is used.



b) Consider the following query written by the stock manager:

```
SELECT stock-id
FROM Stock s1
WHERE quantity = (SELECT avg(s2.quantity)
                  FROM stock s2 WHERE
                  s2.type = s1.type);
```

Explain the semantics of the query and show how you could rewrite it to improve its execution performance. Justify.

Answer: The query above computes the stock-id from the materials whose quantity is equal to the average quantity of the material with the same type as their one.
One option to improve the execution of this query is to create a Temporary table with the average quantity per type.

```
INSERT INTO Temp
SELECT AVG(quantity) as avgQuantity, type FROM Stock
GROUP by  type;
SELECT stock-id
FROM Stock, Temp
WHERE quantity = avgQuantity  AND Stock.type = Temp.type
```

c) Suppose the manager wants to know, throughout the day, the name, and performance evaluation of the nurses along with their schedule slots. Suggest a schema of the database

that is alternative to the one given above that could lead to better performance for the execution of this query. Justify.

Answer:

A possible option to improve the performance for the execution of this query is to create a materialized view where we can find all the information required by the query.

Nurses(professional-id, name, performance-evaluation, schedule-id, slot-start, slot-end)

Question 7 – Miscellaneous

a) Explain with your own words the purpose of the CHECK operators proposed in the paper "Robust Query Processing through Progressive Optimization", by Volker Markl et al and published at SIGMOD 2004. Give an example of their use.

Answer:

CHECK stands for Checkpoint. These operators compare the cardinality estimates of the optimizer with the current cardinalities. The CHECK operator has a condition that establishes the range of cardinality values for which a plan is considered to be valid. If this condition is not satisfied , a re-optimization procedure is triggered.

b) The paper "C-store: A Column-oriented DBMS" (SIGMOD 2005) uses the concept of **projection** in a way that is different from the way it is typically used. Explain why and give an example.

Answer:

A projection is a group of columns sorted on the same attribute. A projection contains one or more attributes from a given table (anchor table) and it may contain a number of attributes from other tables, if there is a foreign key relationship between those tables and the anchor table.
This kind of data organization enables to compress data and thus to support many column sort-orders that open optimization opportunities.

c) Pig Latin is a dataflow language proposed in 2008 by Olston et al. Explain with your own words the meaning of **dataflow language** and mention for which type of database applications this type of language is appropriate.

Answer:

The dataflow language allows the user to specify, like in a programming language, a sequence of steps where each step corresponds to a single, high-level transformation. Its features include support for a flexible and nested data model, user-defined functions and the ability to process input sets without any schema information.

d) Explain how the recovery manager ensures the properties of atomicity and durability of transactions.

Answer: The recovery manager ensures atomicity by storing UNDO information in a log file that enables to undo the updates of transactions that are active when a crash occurs. It ensures durability by storing REDO information in the log that enables to repeat some updates performed by transactions that committed before a crash occurs.

e) Give an example of a heuristic that an optimizer of a relational database management system can use to reduce the number of execution plans generated for a given query (and explain it).

Answer:

Push-down selections: when a relational query expression involves a join and a selection that only applies to attributes of one of the relations involved in the join, it is typically less expensive to compute first the selection and then the join, because many tuples may be filtered out by the selection. So the optimizer must directly choose that query evaluation plan without computing the one that applies the join and then the selection.

Another heuristic: when a query expression involves more than one join, the optimizer may choose to directly generate left-deep plans where the right hand-side input for each join is a base relation and not the result of an intermediate join. This type of plan is convenient for pipeline evaluation and can thus lead to a less expensive plan than bushy trees.

Submitting the project

Prepare a document containing the answers to the exercises above. Identify clearly your group number and names at the top of the first page.
Save the document as a single PDF file and submit it to the Fénix system. The deadline is 1/6, 23H55. **We will not accept deliveries through email.**