Databases Autumn 2025

Hand-In Exercise 4

November 22, 2025

Aiysha Frutiger
Jannick Seper
Luis Tritschler

Total Points	

Task	**Points**

Aiysha Frutiger
Jannick Seper
Luis Tritschler

Task 1

a)

```
CREATE FUNCTION checkBalance (cid IN NUMBER)
RETURN VARCHAR2
IS
        balanceCalculated NUMBER(10,2);
        balanceSaved NUMBER(10,2);
BEGIN
        SELECT SUM(amount)
        INTO balanceCalculated
        FROM BankTransaction
        WHERE customer_id = cid;
        SELECT balance
        INTO balanceSaved
        FROM Customer
        WHERE customer_id = cid;

        IF balanceCalculated = balanceSaved THEN
                RETURN 'Consistent';
        ELSE
                RETURN 'Inconsistent';
        END IF;
END;
```

b)

```
CREATE PROCEDURE checkAllBalances (consistent OUT VARCHAR2)
IS
        inconsistent BOOLEAN := FALSE;
        CURSOR c IS SELECT customer_id FROM Customer;
BEGIN
        FOR record IN c LOOP
                IF check_balance(record.customer_id) =
                                        'Inconsistent' THEN
                        inconsistent := TRUE;
                END IF;
        END LOOP;

        IF inconsistent THEN
                consistent := 'Inconsistent';
        ELSE
                consistent := 'Consistent';
        END IF;
END;
```

c)

```
CREATE PROCEDURE checkReceive (
        cid                     IN NUMBER,
        amount                  IN NUMBER(10,2),
        description             IN VARCHAR2)
IS
        customerExists          NUMBER;
        customerNotExists       EXCEPTION;
        maxTransaction          BankTransaction.transaction_id%TYPE;
BEGIN
        SELECT COUNT(*)
        INTO customerExists
        FROM Customer
        WHERE customer_id = cid;

        IF customerExists = 0 THEN RAISE customerNotExists;
        END IF;

        UPDATE Customer
        SET balance = balance + amount
        WHERE customer_id = cid;

        SELECT MAX(transaction_id)
        INTO maxTransaction
        FROM BankTransaction;

        INSERT INTO BankTransaction
        VALUES (maxTransaction + 1, cid, SYSDATE,
                                                amount, description);
        COMMIT;

EXCEPTION
        WHEN customerNotExists THEN
                ROLLBACK;
        WHEN OTHERS THEN
                ROLLBACK;
END;
```

Aiysha Frutiger
Jannick Seper
Luis Tritschler

Task 2

a)
We first determine how many tuples fit on one data page and then how many pages are required per relation. The page size is 4096 B, of which 96 B are reserved for the page header. Thus, the usable space per page is

$$4096\,\mathrm{B} - 96\,\mathrm{B} = 4000\,\mathrm{B}.$$

Since data pages are filled only up to 90%, the effective capacity for tuples is

$$0.9 \cdot 4000\,\mathrm{B} = 3600\,\mathrm{B}.$$

Each tuple additionally requires one slot array pointer of 6 B. Hence, the effective record size is

$$\text{record size} = \text{tuple size} + 6\,\mathrm{B}.$$

The number of tuples per page is then

$$\text{tuples per page} = \left\lfloor \frac{3600}{\text{record size}} \right\rfloor,$$

and the number of pages is

$$\text{pages} = \left\lceil \frac{\#\text{tuples}}{\text{tuples per page}} \right\rceil.$$

Movie Average tuple size: 100 B, record size: $100 + 6 = 106$ B.

$$\text{tuples per page} = \left\lfloor \frac{3600}{106} \right\rfloor = 33, \qquad \text{pages} = \left\lceil \frac{1000}{33} \right\rceil = 31.$$

Scene Average tuple size: 50 B, record size: $50 + 6 = 56$ B.

$$\text{tuples per page} = \left\lfloor \frac{3600}{56} \right\rfloor = 64, \qquad \text{pages} = \left\lceil \frac{10000}{64} \right\rceil = 157.$$

Person Average tuple size: 40 B, record size: $40 + 6 = 46$ B.

$$\text{tuples per page} = \left\lfloor \frac{3600}{46} \right\rfloor = 78, \qquad \text{pages} = \left\lceil \frac{6000}{78} \right\rceil = 77.$$

Actor Same tuple size as `Person`, hence same record size: $40 + 6 = 46$ B.

$$\text{tuples per page} = \left\lfloor \frac{3600}{46} \right\rfloor = 78, \qquad \text{pages} = \left\lceil \frac{30000}{78} \right\rceil = 385.$$

Relation	Tuples per page	Number of pages
`Movie`	33	31
`Scene`	64	157
`Person`	78	77
`Actor`	78	385

b)
The page size is 4096 B with a page header of 96 B, thus

$$4096\,\mathrm{B} - 96\,\mathrm{B} = 4000\,\mathrm{B}$$

are available per page. Index pages are filled up to 70%, hence the effective capacity is

$$C = 0.7 \cdot 4000\,\mathrm{B} = 2800\,\mathrm{B}.$$

Pointers in the B^+-tree and tuple identifiers (TIDs) have a length of 6 B, data type `NUMBER` has 10 B, and the average `MovieTitle` length is 90 B.

MovieIDIdx Key size $k = 10\,\mathrm{B}$. In a leaf node, one entry consists of (key, TID) with size $10 + 1 \cdot 6 = 16\,\mathrm{B}$. We reserve 2 pointers of 6 B on each leaf page (e.g. sibling pointers):

$$t_{\text{leaf}} = \left\lfloor \frac{(4096 - 96 - (2 \cdot 6)) \cdot 0.7}{10 + (1 \cdot 6)} \right\rfloor = 174.$$

For inner nodes, one entry consists of (key, child pointer) of size $10 + 6 = 16\,\mathrm{B}$, plus an additional pointer of 6 B:

$$e_i = \left\lfloor \frac{(4096 - 96) \cdot 0.7 - 6}{10 + 6} \right\rfloor = 174.$$

With 1000 movies, the number of leaf nodes is

$$n_{\text{leaf}} = \left\lceil \frac{1000}{174} \right\rceil = 6.$$

An inner node can point to $e_i + 1 = 175$ children, thus a single root node suffices:

$$n_{\text{inner}} = \left\lceil \frac{6}{175} \right\rceil = 1.$$

The total index size is

$$i_{\text{size}} = (6 + 1) \cdot 4 = 28\,\mathrm{KiB}.$$

MovieTitleIdx Key size $k = 90\,\mathrm{B}$. Leaf entry size: $90 + 1 \cdot 6 = 96\,\mathrm{B}$:

$$t_{\text{leaf}} = \left\lfloor \frac{(4096 - 96 - (2 \cdot 6)) \cdot 0.7}{90 + (1 \cdot 6)} \right\rfloor = 29.$$

Inner node:

$$e_i = \left\lfloor \frac{(4096 - 96) \cdot 0.7 - 6}{90 + 6} \right\rfloor = 29.$$

With 1000 movies:

$$n_{\text{leaf}} = \left\lceil \frac{1000}{29} \right\rceil = 35.$$

Each inner node can have up to $e_i + 1 = 30$ children, so the number of inner nodes one level above the leaves is

$$n_{\text{inner1}} = \left\lceil \frac{35}{30} \right\rceil = 2,$$

and with one root node above:

$$n_{\text{inner2}} = \left\lceil \frac{2}{30} \right\rceil = 1, \qquad n_{\text{inner}} = 2 + 1 = 3.$$

Thus, the total index size is

$$i_{\text{size}} = (35 + 3) \cdot 4 = 152\,\text{KiB}.$$

ActorIdx There are 30000 tuples in `Actor` and 6000 tuples in `Person`, so on average there are

$$r_k = \frac{30000}{6000} = 5$$

`Actor` entries per `PID`. In the leaf nodes of `ActorIdx`, we store one entry per distinct key `PID`, together with a list of r_k TIDs. Hence, the average leaf entry size is

$$10 + r_k \cdot 6 = 10 + 5 \cdot 6 = 40\,\text{B}.$$

We again reserve 2 pointers of 6 B per leaf page:

$$t_{\text{leaf}} = \left\lfloor \frac{(4096 - 96 - (2 \cdot 6)) \cdot 0.7}{10 + (5 \cdot 6)} \right\rfloor = 69.$$

For inner nodes, we only store (key, child pointer):

$$e_i = \left\lfloor \frac{(4096 - 96) \cdot 0.7 - 6}{10 + 6} \right\rfloor = 174.$$

We have 6000 distinct keys (PIDs), hence

$$n_{\text{leaf}} = \left\lceil \frac{6000}{69} \right\rceil = 87.$$

An inner node can point to up to $e_i + 1 = 175$ children, so

$$n_{\text{inner}} = \left\lceil \frac{87}{175} \right\rceil = 1.$$

The total index size is therefore

$$i_{\text{size}} = (87 + 1) \cdot 4 = 352\,\text{KiB}.$$

Index	Entries per leaf nodes	Entries per inner nodes	Number of leaf nodes	Number of inner nodes	Index Size [KiB]
`MovieIDIdx`	174	174	6	1	28
`MovieTitleIdx`	29	29	35	3	152
`ActorIdx`	69	174	87	1	352

c)

```
SELECT * FROM Movie WHERE MovieID = 4711;

SELECT * FROM Movie WHERE MovieTitle = 'Opelgang';

SELECT * FROM Actor WHERE PID = 1199;
```

Movie.MovieID For `MovieIDIdx` we have $n_{\text{leaf}} = 6$ leaf nodes and a maximum fanout of 175 children per inner node. Hence, the height of the B^+-tree (including root and leaves) is

$$h = \lceil \log_{175} 6 \rceil + 1 = 2.$$

Thus we access $h = 2$ index pages plus one data page in `Movie`:

$$\text{total page accesses} = 2 + 1 = 3.$$

Movie.MovieTitle For `MovieTitleIdx` we have $n_{\text{leaf}} = 35$ leaf nodes and a fanout of 30 children per inner node. The height is

$$h = \lceil \log_{30} 35 \rceil + 1 = 3.$$

Again, we access one data page in `Movie`:

$$\text{total page accesses} = 3 + 1 = 4.$$

Actor.PID For `ActorIdx` we have $n_{\text{leaf}} = 87$ leaf nodes and a fanout of 175 children per inner node. The height is

$$h = \lceil \log_{175} 87 \rceil + 1 = 2.$$

On average there are $r_k = 5$ `Actor` tuples per `PID`, so we expect to access 5 data pages in `Actor`:

$$\text{total page accesses} = 2 + 5 = 7.$$

Relation	Attribute	Page accesses
`Movie`	`MovieID`	3
`Movie`	`MovieTitle`	4
`Actor`	`PID`	7