Analysis of Error ORA-01002: fetch out of sequence

The error **ORA-01002: fetch out of sequence** is occurring due to the **COMMIT** being executed inside the cursor loop. This is a common problem when using FOR UPDATE cursors.

🔍 Cause of the Problem

When you execute a `COMMIT` inside the loop:

1. The cursor `c_emp_update` was opened with `FOR UPDATE NOWAIT`
2. The `COMMIT` releases all locks held by the cursor
3. The cursor becomes inconsistent
4. On the next iteration, Oracle cannot fetch correctly

✅ Recommended Solutions

Solution 1: Remove COMMIT from Loop

```
DECLARE
  CURSOR c_emp_update IS
    SELECT e.employee_id, e.salary, e.commission_pct, j.job_id
    FROM employees e
    JOIN jobs j ON e.job_id = j.job_id
    WHERE e.department_id = 50
    FOR UPDATE OF e.salary, e.commission_pct NOWAIT;

  v_avg_salary NUMBER;
  v_counter NUMBER := 0;

BEGIN
  FOR emp_rec IN c_emp_update LOOP
    -- Get average salary for the job
    SELECT AVG(salary) INTO v_avg_salary
    FROM employees
    WHERE job_id = emp_rec.job_id;

    -- Update salary if below average
    IF emp_rec.salary < v_avg_salary THEN
      UPDATE employees
      SET salary = salary * 1.05
      WHERE CURRENT OF c_emp_update;
```

```
      DBMS_OUTPUT.PUT_LINE('Updated salary for employee ' ||
emp_rec.employee_id);
    END IF;

    -- Set commission if null
    IF emp_rec.commission_pct IS NULL THEN
      UPDATE employees
      SET commission_pct = 0.1
      WHERE CURRENT OF c_emp_update;

      DBMS_OUTPUT.PUT_LINE('Updated commission for employee ' ||
emp_rec.employee_id);
    END IF;

    v_counter := v_counter + 1;
  END LOOP;

  -- Commit only at the end
  COMMIT;
  DBMS_OUTPUT.PUT_LINE('Final commit. Total updates: ' || v_counter);

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No employees found in the specified department.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
    ROLLBACK;
END;
/
```

Solution 2: Batch Processing (Recommended for Large Volumes)

```
DECLARE
  TYPE emp_array_type IS TABLE OF employees.employee_id%TYPE;
  v_emp_ids emp_array_type;
  v_batch_size CONSTANT NUMBER := 5;
  v_total_count NUMBER := 0;
  v_processed NUMBER := 0;

BEGIN
  -- First, collect all IDs that need to be processed
  SELECT employee_id
  BULK COLLECT INTO v_emp_ids
  FROM employees e
```

```
  JOIN jobs j ON e.job_id = j.job_id
  WHERE e.department_id = 50;

  v_total_count := v_emp_ids.COUNT;
  DBMS_OUTPUT.PUT_LINE('Total employees to process: ' || v_total_count);

  -- Process in batches
  FOR i IN 1..v_emp_ids.COUNT LOOP
    DECLARE
      CURSOR c_emp_update IS
        SELECT e.employee_id, e.salary, e.commission_pct, j.job_id
        FROM employees e
        JOIN jobs j ON e.job_id = j.job_id
        WHERE e.employee_id = v_emp_ids(i)
        FOR UPDATE OF e.salary, e.commission_pct NOWAIT;

      v_avg_salary NUMBER;
    BEGIN
      FOR emp_rec IN c_emp_update LOOP
        -- Get average salary for the job
        SELECT AVG(salary) INTO v_avg_salary
        FROM employees
        WHERE job_id = emp_rec.job_id;

        -- Update salary if below average
        IF emp_rec.salary < v_avg_salary THEN
          UPDATE employees
          SET salary = salary * 1.05
          WHERE CURRENT OF c_emp_update;

          DBMS_OUTPUT.PUT_LINE('Updated salary for employee ' ||
emp_rec.employee_id);
        END IF;

        -- Set commission if null
        IF emp_rec.commission_pct IS NULL THEN
          UPDATE employees
          SET commission_pct = 0.1
          WHERE CURRENT OF c_emp_update;

          DBMS_OUTPUT.PUT_LINE('Updated commission for employee ' ||
emp_rec.employee_id);
        END IF;

        v_processed := v_processed + 1;
      END LOOP;
```

```
      -- Commit every batch
      IF MOD(i, v_batch_size) = 0 OR i = v_emp_ids.COUNT THEN
        COMMIT;
        DBMS_OUTPUT.PUT_LINE('Committed batch. Processed: ' || v_processed ||
'/' || v_total_count);
      END IF;
    END;
  END LOOP;

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No employees found in the specified department.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
    ROLLBACK;
END;
/
```

Solution 3: Use BULK COLLECT (More Efficient)

```
DECLARE
  TYPE emp_rec_type IS RECORD (
    employee_id employees.employee_id%TYPE,
    salary employees.salary%TYPE,
    commission_pct employees.commission_pct%TYPE,
    job_id employees.job_id%TYPE
  );

  TYPE emp_array_type IS TABLE OF emp_rec_type;
  v_employees emp_array_type;
  v_avg_salary NUMBER;
  v_batch_size CONSTANT NUMBER := 100;

BEGIN
  -- Collect all data at once
  SELECT e.employee_id, e.salary, e.commission_pct, e.job_id
  BULK COLLECT INTO v_employees
  FROM employees e
  JOIN jobs j ON e.job_id = j.job_id
  WHERE e.department_id = 50;

  -- Process the data
  FOR i IN 1..v_employees.COUNT LOOP
    -- Get average salary for the job
```

```
    SELECT AVG(salary) INTO v_avg_salary
    FROM employees
    WHERE job_id = v_employees(i).job_id;

    -- Update salary if below average
    IF v_employees(i).salary < v_avg_salary THEN
      UPDATE employees
      SET salary = salary * 1.05
      WHERE employee_id = v_employees(i).employee_id;

      DBMS_OUTPUT.PUT_LINE('Updated salary for employee ' ||
v_employees(i).employee_id);
    END IF;

    -- Set commission if null
    IF v_employees(i).commission_pct IS NULL THEN
      UPDATE employees
      SET commission_pct = 0.1
      WHERE employee_id = v_employees(i).employee_id;

      DBMS_OUTPUT.PUT_LINE('Updated commission for employee ' ||
v_employees(i).employee_id);
    END IF;

    -- Commit in batches
    IF MOD(i, v_batch_size) = 0 THEN
      COMMIT;
      DBMS_OUTPUT.PUT_LINE('Committed batch of ' || v_batch_size || '
updates');
    END IF;
  END LOOP;

  -- Final commit
  COMMIT;
  DBMS_OUTPUT.PUT_LINE('Final commit. Total processed: ' ||
v_employees.COUNT);

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No employees found in the specified department.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
    ROLLBACK;
END;
/
```

📋 Solution Summary

Solution	Pros	Cons	Recommendation
Solution 1	Simple, fixes the error	No batch control	Small volumes
Solution 2	Batch control, safer	More complex	Medium volumes
Solution 3	More efficient, better performance	Uses more memory	Large volumes

Solution in Plain English

What's Causing the Error?

The error **ORA-01002: fetch out of sequence** happens because you're doing a `COMMIT` inside the cursor loop. Here's what's going wrong:

1. Your cursor opens and **locks** the rows with `FOR UPDATE`
2. When you do `COMMIT` inside the loop, Oracle **releases all the locks**
3. The cursor gets **confused** and doesn't know where it was
4. Next time it tries to get the next row, it fails

The Simple Fix

Just remove the COMMIT from inside the loop!

Instead of committing every 5 records, only commit at the very end:

```
-- Remove this part from inside the loop:
-- IF MOD(v_counter, v_batch_size) = 0 THEN
--   COMMIT;
-- END IF;

-- Keep only the final commit at the end
COMMIT;
```

Why This Happens

Think of it like this:

- The cursor is like holding a **bookmark** in a book while someone is **holding the pages steady**
- `FOR UPDATE` = someone holding the pages steady
- `COMMIT` = telling that person to let go

- When they let go, your bookmark becomes useless because the pages might move

If You Really Need Batch Processing

If you're dealing with lots of data and need to commit in batches, you have two better options:

Option 1: Process One Record at a Time

- Get a list of all employee IDs first
- Loop through the list
- For each ID, open a small cursor, update, close cursor
- Commit every few records

Option 2: Load Everything into Memory First

- Get all the data you need in one go
- Process it in memory
- Update the database without cursors
- Commit in batches

Bottom Line

The easiest fix is: **Don't commit inside a FOR UPDATE cursor loop.** Just process all your records and commit once at the end. This will solve your error immediately.

💡 Important Tip

Never execute COMMIT inside a loop that uses a FOR UPDATE cursor, as this will always cause the ORA-01002 error. If you need transactional control, use one of the alternative approaches presented above.