

ORACLE PL/SQL · PERTEMUAN 11 / 14

Cursor & Transaction PL/SQL

Membaca banyak baris dengan aman dan menentukan batas transaksi.

Instalasi & SQL → Mini Project → PL/SQL → Final Project · Oracle XE lokal atau FreeSQL dengan login

Versi PDF materi pertemuan 11. Contoh dan tabel dapat dibaca di PDF. Kuis dapat dijalankan di halaman web: [Buka materi daring](#).

Tujuan pembelajaran

1. Membedakan implicit cursor, explicit cursor, dan SELECT INTO.
2. Menjalankan OPEN, FETCH, CLOSE serta cursor FOR loop.
3. Menggunakan cursor berparameter dan empat atribut cursor secara tepat.
4. Mengendalikan COMMIT, ROLLBACK, serta SAVEPOINT tanpa mengubah seed permanen.

Pokok bahasan / praktikum

- Implicit cursor SQL
- Explicit cursor: DECLARE OPEN FETCH CLOSE
- Cursor FOR loop
- Parameterized cursor
- %FOUND %NOTFOUND %ROWCOUNT %ISOPEN
- SELECT INTO dan exception
- Transaction: COMMIT ROLLBACK SAVEPOINT
- DDL implicit commit

Prasyarat: Modul 02/03 dan blok PL/SQL, IF, LOOP, exception modul 09/10. Tidak memerlukan sequence atau package modul 13.

Pilih salah satu lingkungan praktikum

Oracle XE: XE lokal dengan SQL Developer: cocok untuk instalasi, koneksi, user/schema, privilege, tablespace serta latihan SQL/PLSQL.

FreeSQL: FreeSQL dengan login: masuk menggunakan akun sendiri dan gunakan schema latihan yang diizinkan untuk SQL/PLSQL. Tidak perlu memasang XE untuk menyelesaikan latihan non-administratif.

Pengumpulan tugas menerima hasil dari salah satu lingkungan. Pastikan objek prasyarat tersedia dan tetap gunakan schema latihan, bukan schema produksi.

AMATI → PREDIKSI → JELASKAN

Mulai dari pertanyaan

1. Bagaimana mencetak semua produk jika SELECT INTO hanya cocok untuk satu baris?
2. Apa yang terjadi bila FETCH tidak lagi memperoleh baris?
3. Apakah menutup cursor sama dengan membatalkan UPDATE?

Uji pemahaman awal

1. Atribut mana yang aman dibaca sesudah explicit cursor ditutup?

- %FOUND
- %ROWCOUNT
- %ISOPEN
- %NOTFOUND

2. Apa akibat query kosong dalam cursor FOR loop?

- NO_DATA_FOUND selalu terjadi
- Tidak ada iterasi
- Cursor tidak ditutup
- Satu record NULL dibuat

3. ROLLBACK TO savepoint melakukan apa?

- Menghapus tabel
- Menyimpan seluruh perubahan
- Mengakhiri sesi
- Membatalkan perubahan setelah penanda

PAHAMI KONSEP · BACA TERSTRUKTUR · TERAPKAN

Teori dan pembahasan

Cursor mengatur pembacaan baris, sedangkan transaksi mengatur keputusan perubahan data. Jangan menyamakan penutupan cursor dengan rollback atau pelepasan lock transaksi.

Konsep inti bab ini

- FETCH dulu, periksa %NOTFOUND , baru proses record.
- Cursor FOR loop mengelola OPEN/FETCH/CLOSE secara otomatis.
- Simpan atribut implicit cursor sebelum statement SQL lain menggantikannya.

Urutan pembahasan

- 01** Lingkungan dan tujuan transaksi
- 02** Implicit cursor dan SELECT INTO
- 03** Siklus explicit cursor
- 04** Empat atribut dan waktu membacanya
- 05** Cursor FOR dan parameter
- 06** SAVEPOINT, COMMIT, dan ROLLBACK

Gambaran konsep dalam satu tabel

Cursor / atribut	Makna	Waktu pemeriksaan
Implicit cursor	Dikelola Oracle untuk statement SQL	Sesudah statement yang relevan
Explicit cursor	Hasil query bernama	<code>OPEN</code> → <code>FETCH</code> → <code>CLOSE</code>
%FOUND / %NOTFOUND	Keberhasilan fetch terakhir	Sesudah <code>FETCH</code>
%ROWCOUNT	Baris yang telah diambil/terdampak	Bukan ukuran hasil sebelum dibaca
%ISOPEN	Status cursor terbuka	Aman untuk memeriksa lifecycle

01 Lingkungan dan tujuan transaksi

Gunakan Oracle XE 21c lokal melalui SQL Developer ATAU Oracle FreeSQL setelah login pada schema sendiri yang diizinkan; bukan dua kewajiban.

Selesaikan setup lima tabel dan seed modul 02/03 sekali, tanpa menghapus data. Aktifkan panel DBMS Output pada koneksi lokal atau output editor FreeSQL. SQL memakai titik koma; slash pada baris sendiri adalah delimiter Run Script klien, bukan sintaks server. Jika editor mengirim satu unit PL/SQL, kirim blok tanpa slash. Seluruh latihan menggunakan objek sendiri, tanpa DBA atau langkah administrasi.

POIN PEMBAHASAN

- Cursor membantu program membaca hasil query per baris, sedangkan transaksi menyatukan perubahan data menjadi satu keputusan.
- Keduanya berbeda: `CLOSE` melepaskan cursor, bukan mengembalikan stok.
- Sebelum percobaan, pastikan tidak ada pekerjaan penting yang belum disimpan di koneksi latihan.
- Gunakan transaksi baru dan catat nilai awal produk 11.

- Contoh sengaja membatalkan perubahan agar penjualan 101–103 tetap tersedia bagi modul berikutnya.

02 Implicit cursor dan SELECT INTO

Oracle mengelola implicit cursor untuk pernyataan SQL yang tidak memakai cursor bernama.

Setelah `UPDATE`, `SQL%ROWCOUNT` menunjukkan jumlah baris yang diubah, `SQL%FOUND` menunjukkan ada baris terdampak, dan `SQL%NOTFOUND` menunjukkan tidak ada baris terdampak. `SQL%ISOPEN` selalu `FALSE` karena Oracle menutup implicit cursor setelah pernyataan selesai. Simpan atribut segera bila hasilnya akan dipakai, sebab SQL berikutnya dapat menggantinya.

POIN PEMBAHASAN

- `SELECT INTO` mengharapkan tepat satu baris.
- Nol baris menghasilkan `NO_DATA_FOUND`; lebih dari satu menghasilkan `TOO_MANY_ROWS`.
- Itu bukan perilaku cursor loop: query kosong pada loop cukup menghasilkan nol iterasi.
- Jangan menganggap pengecualian otomatis membatalkan seluruh pekerjaan pemanggil.
- Tangani kondisi bisnis yang dikenali, lalu serahkan keputusan transaksi kepada pemiliknya.

03 Siklus explicit cursor

Explicit cursor dideklarasikan dengan nama dan query, kemudian `OPEN` menjalankan query serta menyiapkan hasilnya.

`FETCH` memindahkan baris berikutnya ke record atau variabel yang cocok, dan `CLOSE` mengakhiri penggunaan cursor. Pola aman adalah `FETCH` dahulu, `EXIT WHEN cursor%NOTFOUND` kemudian, baru memproses record. Membaca record sebelum pengecekan dapat mencetak nilai lama saat hasil telah habis.

POIN PEMBAHASAN

- Gunakan `%ROWTYPE` agar tipe record mengikuti hasil cursor.
- Bila pemrosesan mengeluarkan exception, tutup cursor yang masih terbuka pada bagian penanganan, lalu `RAISE` kembali.
- Membuka cursor yang sudah terbuka dan `FETCH` pada cursor tertutup adalah kesalahan, bukan cara memulai ulang pencarian.
- Setelah ditutup, cursor dapat dibuka kembali untuk menjalankan query baru.

04 Empat atribut dan waktu membacanya

`%ISOPEN` dapat diperiksa kapan saja untuk mengetahui status cursor.

Setelah `OPEN` tetapi sebelum `FETCH` pertama, `%FOUND` dan `%NOTFOUND` bernilai `NULL`; `%ROWCOUNT` masih nol. `FETCH` yang berhasil membuat `%FOUND` `TRUE` dan menambah `%ROWCOUNT`. `FETCH` yang gagal membuat `%NOTFOUND` `TRUE` tanpa menambah hitungan.

POIN PEMBAHASAN

- Setelah `CLOSE`, hanya `%ISOPEN` yang aman dibaca.
- Atribut lainnya pada explicit cursor tertutup menghasilkan `INVALID_CURSOR`.
- Karena itu contoh mencetak hitungan sebelum `CLOSE`.
- Jangan menyamakan `%ROWCOUNT` dengan seluruh ukuran hasil sebelum loop selesai: nilainya menghitung baris yang telah diambil, bukan jumlah yang belum dibaca.

05 Cursor FOR dan parameter

Cursor FOR loop otomatis mendeklarasikan record, membuka cursor, mengambil baris, dan menutupnya ketika selesai, keluar dari loop, atau terjadi exception.

Record hanya berlaku di dalam loop. Pakai alias pada kolom ekspresi agar nama field jelas. Untuk laporan singkat, `FOR r IN (SELECT ...)` mengurangi pekerjaan administratif tanpa menyembunyikan logika query.

POIN PEMBAHASAN

- Cursor berparameter memisahkan query dari nilai filter.
- Cursor `c_produk(p_min NUMBER)` menerima batas stok ketika dipanggil.
- Nama parameter jangan sama dengan kolom agar kondisi tidak berubah menjadi perbandingan kolom dengan dirinya sendiri.
- Menjalankan `c_produk(20)` pada seed menghasilkan produk 11, 12, dan 13, bukan produk 14 yang stoknya nol.

06 SAVEPOINT, COMMIT, dan ROLLBACK

COMMIT menyimpan seluruh perubahan transaksi koneksi dan melepaskan lock.

ROLLBACK membatalkan seluruh perubahan yang belum disimpan. **SAVEPOINT** memberi penanda; **ROLLBACK TO** penanda hanya mengembalikan pekerjaan setelah penanda tanpa mengakhiri transaksi. Tidak ada **BEGIN TRANSACTION** ala SQL Server: DML memulai pekerjaan transaksi sesuai konteks Oracle.

POIN PEMBAHASAN

- DDL dapat melakukan implicit commit, sehingga **CREATE TABLE** atau **CREATE PROCEDURE** harus dipisahkan dari percobaan rollback.
- DDL bukan bagian yang bisa dipulihkan dengan savepoint latihan ini.
- **NULL** berarti tidak diketahui; string kosong diperlakukan sebagai **NULL** di Oracle.
- Gunakan **IS NULL**, bukan **= NULL**, terutama ketika membaca kota Citra atau pelanggan anonim.
- Nyatakan hasil sebagai prediksi sampai benar-benar menjalankan Oracle dan merekam output sendiri.

Siklus explicit cursor

- 1 Deklarasikan query
- 2 `OPEN` cursor
- 3 `FETCH` satu baris
- 4 Periksa NOTFOUND lalu proses
- 5 `CLOSE` dan tentukan transaksi

Gunakan alur ini sebagai panduan saat mengikuti praktikum; bukti hasil tetap berasal dari eksekusi Oracle Anda.

AMATI DATA · IKUTI LANGKAH · PAHAM HASIL

Contoh dan tabel pembahasan

Contoh berikut dibaca langsung, tanpa parameter atau tombol simulasi. Nilai adalah **prediksi ilustratif dari data acuan**, bukan hasil eksekusi pada akun Oracle Anda.

Jejak OPEN dan FETCH pelanggan

Cursor berurutan menurut id_pelanggan membaca Ayu, Bima dan Citra.

Tahap	%ROWCOUNT	%FOUND	%NOTFOUND	Tindakan
Setelah <code>OPEN</code>	0	<code>NULL</code>	<code>NULL</code>	Belum ada fetch
<code>FETCH 1</code>	1	<code>TRUE</code>	<code>FALSE</code>	Ayu
<code>FETCH 2</code>	2	<code>TRUE</code>	<code>FALSE</code>	Bima
<code>FETCH 3</code>	3	<code>TRUE</code>	<code>FALSE</code>	Citra

Tahap	%ROWCOUNT	%FOUND	%NOTFOUND	Tindakan
FETCH setelah baris terakhir	3	FALSE	TRUE	Hentikan loop; jangan proses record

Cara membaca contoh

1. `OPEN` menyiapkan hasil; sebelum `FETCH`, `%FOUND` dan `%NOTFOUND` belum mempunyai nilai `TRUE/FALSE`.
2. Setiap `FETCH` yang berhasil menaikkan `%ROWCOUNT` dan memperbolehkan record diproses.
3. `FETCH` setelah baris terakhir mengaktifkan `%NOTFOUND` tanpa menaikkan `%ROWCOUNT`. Keluar lalu `CLOSE`.

Yang perlu dipahami: Menutup cursor tidak sama dengan melakukan `COMMIT`. Keputusan transaksi tetap terpisah.

Video pembelajaran

Video utama — fokus bab ini

Implicit/explicit cursor, parameterized cursor dan proses fetch Oracle PL/SQL.

Oracle PLSQL - CURSOR (Implicit and Explicit)

Ora Trainings · [Buka video utama di YouTube](#)

Batas cakupan dan panduan menonton: Video ini khusus cursor. Gabungkan dengan kontrol transaksi `COMMIT/ROLLBACK/SAVEPOINT` pada praktikum11 dan pengantar transaksi pertemuan03; jangan menempatkan `COMMIT` otomatis di setiap fetch.

Video merupakan pendamping, bukan pengganti keseluruhan bab. Pemeriksaan relevansi memakai transkrip yang tersedia, deskripsi pembuat dan bab waktu; bukan klaim seluruh durasi telah ditonton. Versi/editor atau bahasa dapat berbeda. Praktikum tetap mengikuti SQL/PLSQL Oracle pada halaman dan pilihan XE atau FreeSQL login. Jika pemutar dibatasi oleh YouTube, gunakan tautan Buka video.

Praktikum Oracle XE atau FreeSQL

Pilih salah satu lingkungan praktikum

Oracle XE: XE lokal dengan SQL Developer: cocok untuk instalasi, koneksi, user/schema, privilege, tablespace serta latihan SQL/PLSQL.

FreeSQL: FreeSQL dengan login: masuk menggunakan akun sendiri dan gunakan schema latihan yang diizinkan untuk SQL/PLSQL. Tidak perlu memasang XE untuk menyelesaikan latihan non-administratif.

Pengumpulan tugas menerima hasil dari salah satu lingkungan. Pastikan objek prasyarat tersedia dan tetap gunakan schema latihan, bukan schema produksi.

Gunakan [FreeSQL dengan login](#) atau SQL Developer yang terhubung ke XE. [Baca panduan dua jalur praktikum](#). Jalankan kode sesuai urutan dan keterangan tiap langkah.

Output contoh adalah prediksi berdasarkan dataset, bukan bukti bahwa kode telah dijalankan pada akun Anda. Periksa koneksi, hak membuat objek, autocommit dan panel DBMS_OUTPUT. Slash pada baris sendiri adalah pemisah mode skrip klien tertentu, bukan statement PL/SQL.

Gunakan tabel pertemuan02 dan data pertemuan03 atau setup index sekali. Periksa keadaan data bila latihan sebelumnya pernah di-commit. Jangan menjalankan ulang CREATE TABLE/INSERT atau menghapus data secara buta.

Langkah 1 — Implicit cursor setelah UPDATE

```
DECLARE
  v_baris PLS_INTEGER;
BEGIN
  SAVEPOINT awal_11;
  UPDATE produk_demo SET stok = stok + 1 WHERE id_produk = 11;
  v_baris := SQL%ROWCOUNT;
  IF SQL%FOUND THEN DBMS_OUTPUT.PUT_LINE('UPDATE menemukan
produk'); END IF;
  IF NOT SQL%NOTFOUND THEN DBMS_OUTPUT.PUT_LINE('Bukan hasil
kosong'); END IF;
  IF NOT SQL%ISOPEN THEN DBMS_OUTPUT.PUT_LINE('Implicit cursor
tertutup'); END IF;
```

```

    DBMS_OUTPUT.PUT_LINE('Baris berubah=' || v_baris);
    ROLLBACK TO awal_11;
EXCEPTION WHEN OTHERS THEN
    ROLLBACK TO awal_11;
    RAISE;
END;
/
SELECT id_produk, stok FROM produk_demo WHERE id_produk = 11;

```

Pada seed: satu baris berubah sementara, kemudian stok kembali 20. Blok merupakan pemanggil latihan; rollback bukan perilaku helper.

Langkah 2 — Explicit cursor lengkap dengan cleanup

```

DECLARE
    CURSOR c_produk IS SELECT id_produk, nama_produk, stok FROM
    produk_demo ORDER BY id_produk;
    r c_produk%ROWTYPE;
BEGIN
    OPEN c_produk;
    LOOP
        FETCH c_produk INTO r;
        EXIT WHEN c_produk%NOTFOUND;
        IF c_produk%FOUND THEN
            DBMS_OUTPUT.PUT_LINE(r.id_produk || ': ' || r.nama_produk || ',
stok=' || r.stok);
        END IF;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('Fetched=' || c_produk%ROWCOUNT);
    CLOSE c_produk;
    IF NOT c_produk%ISOPEN THEN DBMS_OUTPUT.PUT_LINE('Cursor telah
ditutup'); END IF;
EXCEPTION WHEN OTHERS THEN
    IF c_produk%ISOPEN THEN CLOSE c_produk; END IF;
    RAISE;
END;
/

```

Prediksi empat produk, Fetched=4. Atribut ROWCOUNT dibaca sebelum CLOSE.

Langkah 3 — Cursor FOR berparameter dan query inline

```

DECLARE
  CURSOR c_produk(p_min NUMBER) IS
    SELECT id_produk, nama_produk, stok FROM produk_demo
    WHERE stok >= p_min ORDER BY id_produk;
BEGIN
  FOR r IN c_produk(20) LOOP
    DBMS_OUTPUT.PUT_LINE(r.nama_produk || ': ' || r.stok);
  END LOOP;
  FOR r IN (SELECT id_pelanggan, nama FROM pelanggan_demo ORDER BY
id_pelanggan) LOOP
    DBMS_OUTPUT.PUT_LINE('Pelanggan ' || r.id_pelanggan || ': ' ||
r.nama);
  END LOOP;
END;
/

```

Loop pertama menampilkan Kopi, Teh, Gula pada seed; loop kedua tiga pelanggan. Cursor ditutup otomatis.

Langkah 4 — SELECT INTO berbeda dari hasil cursor kosong

```

DECLARE
  v_nama pelanggan_demo.nama%TYPE;
BEGIN
  BEGIN
    SELECT nama INTO v_nama FROM pelanggan_demo WHERE id_pelanggan =
9999;
    EXCEPTION WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('SELECT
INTO: tidak ditemukan');
  END;
  FOR r IN (SELECT nama FROM pelanggan_demo WHERE id_pelanggan =
9999) LOOP
    DBMS_OUTPUT.PUT_LINE(r.nama);
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('Loop kosong selesai tanpa NO_DATA_FOUND');
END;
/

```

Pelanggan 9999 tidak ada pada seed. Tidak ada perubahan data.

Langkah 5 — Pemilik transaksi: savepoint, rollback penuh, commit

```
-- Jalankan di koneksi latihan yang tidak memiliki pekerjaan tertunda.  
SAVEPOINT awal_transaksi_11;  
UPDATE produk_demo SET stok = stok + 2 WHERE id_produk = 12;  
SAVEPOINT sesudah_teh_11;  
UPDATE produk_demo SET stok = stok + 3 WHERE id_produk = 13;  
ROLLBACK TO sesudah_teh_11;  
SELECT id_produk, stok FROM produk_demo WHERE id_produk IN (12,13)  
ORDER BY id_produk;  
ROLLBACK;  
SELECT id_produk, stok FROM produk_demo WHERE id_produk IN (12,13)  
ORDER BY id_produk;  
-- Latihan COMMIT netral: tidak mengubah nilai seed.  
UPDATE produk_demo SET stok = stok WHERE id_produk = 12;  
COMMIT;
```

Sesudah rollback parsial, seed Teh menjadi 32 sementara Gula 40. ROLLBACK penuh mengembalikan 30 dan 40. COMMIT terakhir menyimpan UPDATE netral; bila trigger audit modul 13 sudah ada, perubahan stok netral tidak diaudit.

Tugas / latihan

1. Pada XE lokal ATAU FreeSQL schema sendiri, buat cursor berparameter batas harga dan tampilkan produk yang memenuhi syarat.
2. Bandingkan loop manual dan FOR dengan filter yang sama; sertakan hasil aktual dan jumlah baris.
3. Buat dua UPDATE dengan SAVEPOINT di antaranya, buktikan rollback parsial dan penuh tanpa mengubah seed permanen.
4. Kumpulkan SQL dan catatan perbedaan SELECT INTO kosong dengan cursor kosong; jangan menjalankan DDL di tengah uji rollback.

Pertanyaan analisis

1. Mengapa pengecekan NOTFOUND harus dilakukan setelah FETCH dan sebelum mencetak record?
2. Bagaimana Anda membuktikan bahwa penutupan cursor tidak menyimpan atau membatalkan DML?

Pengumpulan

Tugas boleh dikerjakan di **Oracle XE lokal atau FreeSQL dengan login**; cukup pilih salah satu. Kumpulkan satu laporan melalui LMS/kanal yang ditetapkan dosen. Cantumkan Nama, NPM, pertemuan dan lingkungan yang dipakai. Sertakan kode SQL/PLSQL sebagai teks, output aktual atau pesan error, serta jawaban analisis.

Jika FreeSQL menyediakan tautan berbagi yang dapat diakses dosen, tautan boleh ditambahkan sebagai bukti pendamping, bukan pengganti hasil eksekusi yang terdokumentasi. Jangan membagikan password, token atau koneksi pribadi.

Hasil yang belum dieksekusi diberi label “prediksi”. Screenshot ilustrasi browser bukan bukti praktikum Oracle.

Rubrik penilaian

- Ketepatan konsep dan rancangan: **20%**
- Kebenaran query/program: **30%**
- Pengujian normal, batas dan gagal: **20%**
- Analisis serta penjelasan hasil: **20%**
- Identitas dan kerapian laporan: **10%**

Referensi resmi

- [Oracle 19c — Cursors Overview](#)
- [Oracle 19c — COMMIT](#)

Panduan XE21c digunakan untuk konteks instalasi lokal; dokumentasi SQL/PLSQL untuk konsep bahasa. Versi FreeSQL/editor dapat berbeda, sehingga periksa fitur dan pesan aktual.