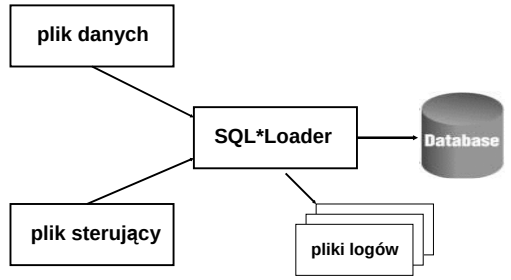
Wczytywanie danych ze źródeł zewnętrznych (1)

- Źródła zewnętrzne
  - bazy danych - heterogeniczność
    - producenci
    - funkcjonalność
    - modele danych
    - protokoły komunikacyjne
  - pliki
    - tekstowe, HTML, XML
    - arkusze kalkulacyjne
    - teksty formatowane edytorami

Wczytywanie danych ze źródeł zewnętrznych (2)

- Protokoły komunikacyjne
  - Net8
  - ODBC/JDBC
  - gateways
- Oprogramowanie Oracle
  - SQL*Loader
  - Warehouse Builder
  - polecenia SQL
    - tabele zewnętrzne (external tables)
    - multitable insert
    - merge

SQL*Loader (1)



SQL*Loader (2)

```
LOAD DATA
INFILE *
INTO TABLE dept
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
(deptno, dname, loc)
BEGINDATA
12,RESEARCH,"SARATOGA"
10,"ACCOUNTING",CLEVELAND
11,"ART",SALEM
13,FINANCE,"BOSTON"
21,"SALES",PHILA.
22,"SALES",ROCHESTER
42,"INT'L","SAN FRAN"
```

```
sqlldr USERID=scott/tiger CONTROL=ulcase1.ctl LOG=ulcase1.log
```


SQL*Loader (3)

```
LOAD DATA
INFILE 'ulcase2.dat'
INTO TABLE emp
(empno          POSITION(01:04)   INTEGER EXTERNAL,
 ename          POSITION(06:15)   CHAR,
 job            POSITION(17:25)   CHAR,
 mgr            POSITION(27:30)   INTEGER EXTERNAL,
 sal            POSITION(32:39)   DECIMAL EXTERNAL,
 comm           POSITION(41:48)   DECIMAL EXTERNAL,
 deptno         POSITION(50:51)   INTEGER EXTERNAL)
```

```
7782 CLARK      MANAGER   7839  2572.50          10
7839 KING       PRESIDENT       5500.00          10
7934 MILLER     CLERK     7782   920.00          10
7566 JONES      MANAGER   7839  3123.75          20
7499 ALLEN      SALESMAN  7698  1600.00   300.00 30
7654 MARTIN     SALESMAN  7698  1312.50  1400.00 30
7658 CHAN       ANALYST   7566  3450.00          20
7654 MARTIN     SALESMAN  7698  1312.50  1400.00 30
```


SQL*Loader (4)

```
LOAD DATA
INFILE *
APPEND
INTO TABLE emp
FIELDS TERMINATED BY "," OPTIONALLY ENCLOSED BY '"'
(empno, ename, job, mgr,
hiredate DATE(20) "DD-Month-YYYY",
sal, comm, deptno CHAR TERMINATED BY ':',
projno,
loadseq  SEQUENCE(MAX,1))
BEGINDATA
   7782, "Clark", "Manager", 7839, 09-June-1981, 2572.50,,  10:101
   7839, "King", "President", , 17-November-1981,5500.00,,10:102
   7934, "Miller", "Clerk", 7782, 23-January-1982, 920.00,, 10:102
   7566, "Jones", "Manager", 7839, 02-April-1981, 3123.75,, 20:101
   7499, "Allen", "Salesman", 7698, 20-February-1981, 1600.00, 300.00, 30:103
   7654, "Martin", "Salesman", 7698, 28-September-1981, 1312.50, 1400.00, 3:103
   7658, "Chan", "Analyst", 7566, 03-May-1982, 3450,,  20:101
```

Wielotabelowy INSERT (1)

➲ **bezwarunkowy INSERT ALL**

```
insert all
into hist_placy values(id_prac, zatrudniony, pensja)
into hist_zatrudnienia
               values(id_prac, zatrudniony, id_zesp)
select id_prac, zatrudniony,
      placa_pod+nvl(placa_dod,0) pensja, id_zesp
from pracownicy
where id_zesp!=50;
```


Wielotabelowy INSERT (2)

➲ **warunkowy INSERT ALL**

- **ten sam rekord może spełniać wiele warunków klauzuli WHEN ⇨ jest wstawiany do wielu tabel**

```
insert all
when id_zesp=10 then
  into hist_zesp10 values(id_prac, zatrudniony, id_zesp)
when id_zesp=20 then
  into hist_zesp20 values(id_prac, zatrudniony, id_zesp)
when placa_pod>600 then
  into hist_placy values(id_prac, zatrudniony, placa_pod)
select id_prac, zatrudniony, id_zesp, placa_pod
from pracownicy;
```


Wielotabelowy INSERT (3)

```
create table sprzedaz_zdenorm
(id_sprzedazy number(6) primary key,
 nr_tygodnia number(2) not null,
 id_produktu number(6) not null,
 ilosc_pon number(4),
 ilosc_wt number(4),
 ilosc_sr number(4),
 ilosc_czw number(4),
 ilosc_pt number(4),
 ilosc_sob number(4),
 ilosc_nie number(4));
```

```
create table sprzedaz
(id_sprzedazy number(6) not null,
 nr_tygodnia number(2) not null,
 id_produktu number(6) not null,
 sprzedaz number(4) not null);
```

```
insert into sprzedaz_zdenorm
    values (1, 21, 100, 40, 45, 35, 120, 310, 290, 20);
insert into sprzedaz_zdenorm
    values (2, 22, 100, 41, 46, 36, 121, 311, 291, 21);
```


Wielotabelowy INSERT (4)

```
insert all
into sprzedaz
     values(id_sprzedazy, nr_tygodnia, id_produktu, ilosc_pon)
into sprzedaz
     values(id_sprzedazy, nr_tygodnia, id_produktu, ilosc_wt)
into sprzedaz
     values(id_sprzedazy, nr_tygodnia, id_produktu, ilosc_sr)
into sprzedaz
     values(id_sprzedazy, nr_tygodnia, id_produktu, ilosc_czw)
into sprzedaz
     values(id_sprzedazy, nr_tygodnia, id_produktu, ilosc_pt)
into sprzedaz
     values(id_sprzedazy, nr_tygodnia, id_produktu, ilosc_sob)
into sprzedaz
     values(id_sprzedazy, nr_tygodnia, id_produktu, ilosc_nie)
select * from sprzedaz_zdenorm;
```


Wielotabelowy INSERT (5)

➲ **warunkowy FIRST INSERT**

- **ten sam rekord może spełniać wiele warunków klauzuli WHEN ⇨ jest wstawiany do tabeli wyłącznie w pierwszej klauzuli WHEN**

```
insert first
when id_zesp=10 then
  into hist_zesp10 values(id_prac, zatrudniony, id_zesp)
when id_zesp=20 then
  into hist_zesp20 values(id_prac, zatrudniony, id_zesp)
when placa_pod>600 then
  into hist_placy values(id_prac, zatrudniony, placa_pod)
select id_prac, zatrudniony, id_zesp, placa_pod
from pracownicy;
```


Wielotabelowy INSERT (6)

```
insert first
  when id_zesp=10 and etat='ASYSTENT' then into p1
  when id_zesp=20 then into p2
  else into p3
select * from pracownicy;
```

Tabele zewnętrzne

- dane przechowywane w plikach tekstowych poza bazą danych
- tylko do odczytu
- nie można tworzyć indeksów
- nie można definiować żadnych ograniczeń integralnościowych
- operowanie przez SELECT
- w momencie wykonywania SELECT dane są dynamicznie wczytywane do tabeli zewnętrznej
- pliki tekstowe (źródło danych dla tabel zewnętrznych) w katalogu wskazywanym przez DIRECTORY

Tabele zewnętrzne - tworzenie (1)

```
create or replace directory
    exttab_sources as 'c:\extTab';
```

```
create table ext_sprzedaz
 (id_sprzedazy number(6),
  nr_tygodnia number(2),
  id_produktu number(6),
  sprzedaz number(4))
organization external
 (type oracle_loader
  default directory exttab_sources
  access parameters opis
 )
parallel 2
reject limit 100;
```

oprogramowanie wczytujące dane z pliku

2 procesy wczytujące równolegle

po 100 rekordach odrzuconych z powodu błędów zapytanie do tabeli zewnętrznej przerywane z błędem

Tabele zewnętrzne - tworzenie (2)

- reject
  - limit *liczba*|unlimited
- type
  - oracle_loader (SQL*Loader)
  - oracle_internal (export/import na razie nie wspierane)

Tabele zewnętrzne - tworzenie (3)

```
create table ext_sprzedaz
 (id_sprzedazy number(6),
  nr_tygodnia number(2),
  id_produktu number(6),
  sprzedaz number(4))
organization external
 (type oracle_loader
  default directory exttab_sources
  access parameters
  ( records delimited by newline
    badfile 'bledy.log'
    logfile 'log.log'
    fields terminated by ';'
    ( id_sprzedazy char(4),
      nr_tygodnia char(4),
      id_produktu char(4),
      sprzedaz char(4)))
    location(exttab_sources1:'sprzedaz.txt'))
parallel 2
reject limit 100;
```

```
create or replace directory
    exttab_sources1 as 'c:\extTab1';
```


Tabele zewnętrzne - klauzule (1)

```
create table ext_sprzedaz
 (id_sprzedazy number(6),
  nr_tygodnia number(2),
  id_produktu number(6),
  sprzedaz number(4))
organization external
 (type oracle_loader
  default directory exttab_sources
  access parameters
  ( records delimited by newline
    load when id_sprzedazy='10'
    logfile 'log.log'
    fields terminated by ';'
    ( id_sprzedazy char(6),
      nr_tygodnia char(4),
      id_produktu char(6),
      sprzedaz char(4)))
    location(exttab_sources1:'sprzedaz.txt'))
parallel 2
reject limit 100;
```

- LOAD WHEN - ograniczenie wczytywanych rekordów
  - klauzula LOAD WHEN kolumna='wartość'
  - warunki ograniczające można nakładać wyłącznie na pierwszą kolumnę w pliku źródłowym

Tabele zewnętrzne - klauzule (2)

- pliki LOGu

```
create table ext_sprzedaz
 (...)
organization external
 (...
  access parameters
  ( ...
    badfile 'bledy.log'
    discardfile 'discard.log'
    logfile 'log.log'
    ...
    location(...));
```

- badfile - rekordy niewczytane z powodu błędów
- discardfile - rekordy odrzucone klauzulą WHEN
- logfile - plik logu z sesji wczytywania

Tabele zewnętrzne - klauzule (3)

- **LOCATION**
- **wiele plików źródłowych, w różnych katalogach**

```
create table ext_sprzedaz
 (...)
organization external
  ...
    location(exttab_sources1:'sprzedaz.txt',
             exttab_sources2:'sprzedaz1.txt',
                             'sprzedaz2.txt')
           )
;
```


Tabele zewnętrzne - klauzule (4)

- **SKIP n**
  - **pomija n pierwszych rekordów z pliku wejściowego, np. nagłówek pliku, nagłówki kolumn**

```
access parameters
 ( records delimited by newline
   skip 2
   ...
```


Tabele zewnętrzne - klauzule (5)

- **MISSING FIELD VALUES ARE NULL**

```
1; 10; 100; 12
2; 11; 100; 22
3; 11; 110; 34
4; 11; 120; 77
5; 12; 100; 33
6; 10; 1; 45
7;
```

bez powyższej klauzuli ten rekord nie zostanie wczytany

```
access parameters
 ( records delimited by newline
   skip 1
   ...
   fields terminated by ';'
   missing field values are null
   ...
```


MERGE (1)

```
create table prac_historia
as select * from pracownicy
where 1=2;
```

```
merge into prac_historia ph
   using pracownicy p
   on (ph.id_prac=p.id_prac)
when matched then
   update set
             ph.placa_pod=p.placa_pod,
             ph.placa_dod=p.placa_dod,
             ph.etat=p.etat
when not matched then
   insert values (p.id_prac, p.nazwisko,
                  p.etat, p.id_szefa,
                  p.zatrudniony,
                  p.placa_pod, p.placa_dod,
                  p.id_zesp);
```

tabela, zapytanie, perspektywa

MERGE (2)

```
create table prac_historia1 as select
nazwisko, etat, placa_pod, id_zesp
from pracownicy where 1=2;
```

```
merge into prac_historia1 ph
   using (select nazwisko, etat, placa_pod, id_zesp
          from pracownicy) p
   on (ph.nazwisko=p.nazwisko and ph.id_zesp=p.id_zesp)
when matched then
   update set
             ph.placa_pod=p.placa_pod,
             ph.etat=p.etat
when not matched then
   insert (ph.id_zesp, ph.placa_pod, ph.etat, ph.nazwisko)
   values (p.id_zesp, p.placa_pod, p.etat, p.nazwisko);
```


CASE (1)

```
case wyrażenie when warunek1 then wynik1
               when warunek2 then wynik2
               ...
               when warunekn then wynikn
               else wynikz
end
```

- **jeżeli wyrażenie spełnia *warunek*$_1$ CASE przyjmuje *wynik*$_1$**
- **w przeciwnym przypadku sprawdzany *warunek*$_2$**
- **jeżeli żaden z warunków nie jest spełniony, wyrażenie przyjmuje *wynik*$_z$ z instrukcji ELSE**
  - **w przypadku braku ELSE, CASE przyjmuje NULL**
- **wszystkie *wyniki* muszą być tego samego typu**
- **jeżeli kilka warunków jest spełnionych, wybierany jest pierwszy**

CASE (2)

```
select nazwisko, etat, placa_pod,
   case etat when 'ADIUNKT' then placa_pod*1.2
             when 'ASYSTENT' then placa_pod*1.1
             when 'PROFESOR' then placa_pod*1.4
   end "po zapowiadanej podwyżce"
from pracownicy;
```

```
select nazwisko, etat, placa_pod,
   case when etat='ADIUNKT' then placa_pod*1.2
        when etat='ASYSTENT' then placa_pod*1.1
        when id_zesp=20 then placa_pod*1.4
   end "po zapowiadanej podwyżce"
from pracownicy;
```


CASE (3)

```
select nazwisko, placa_pod +
   (case when placa_dod is null then 0
         else placa_dod
   end) pensja
from pracownicy;
```

CASE w wyrażeniu

```
select c.zespol
from (select case id_zesp when 10 then 'Administracja'
                          when 20 then 'Systemy rozproszone'
                          else 'Inny'
            end zespol
      from zespoly) c;
```

CASE w klauzuli FROM

Funkcja COALESCE

- wartością funkcji jest pierwsza niepusta wartość|wyrażenie z listy

```
coalesce (wyrażenie1, wyrażenie2, ..., wyrażenien)
```

```
select coalesce(placa_dod, placa_pod, 0) from pracownicy;
```

- jeżeli wartość *placa_dod* nie jest pusta, funkcja zwróci jej wartość
- jeżeli *placa_dod* jest pusta, a *placa_pod* nie jest pusta, funkcja zwróci wartość *placa_pod*
- jeżeli obie płace są puste, funkcja zwróci 0

Funkcje tablicowe

- Funkcje zwracające kolekcje rekordów
- Wykorzystywane w zapytaniach jak tabele/perspektywy
- Zastosowanie: złożone transformacje danych, np. w procesie ETL
- Cechy
  - udostępniają rekord już w momencie jego wyznaczenia w funkcji
  - przetwarzanie potokowe
  - możliwość zrównoleglenia przetwarzania
  - wykorzystują obiektowy model danych
    - złożone typy danych użytkownika
    - kolekcje

Funkcje tablicowe - przykład

```
create or replace type employee_T as object
( name varchar2(20),
  job varchar2(15)
)
/
```

```
create or replace type
employee_Collection as
table of employee_T
/
```

```
CREATE OR REPLACE FUNCTION F_empl
RETURN employee_Collection
PIPELINED AS
  cursor c_emp is select ename, job from emp;
  rec_emp c_emp%ROWTYPE;
  rec_out employee_T := employee_T(null, null);
BEGIN
  open c_emp;
  loop
    fetch c_emp into rec_emp;
    exit when c_emp%NOTFOUND;
    rec_out.name:=rec_emp.ename;
    rec_out.job:=rec_emp.job;
    PIPE ROW (rec_out);
  end loop;
  return;
END;
/
```

```
select * from TABLE(F_empl());
```