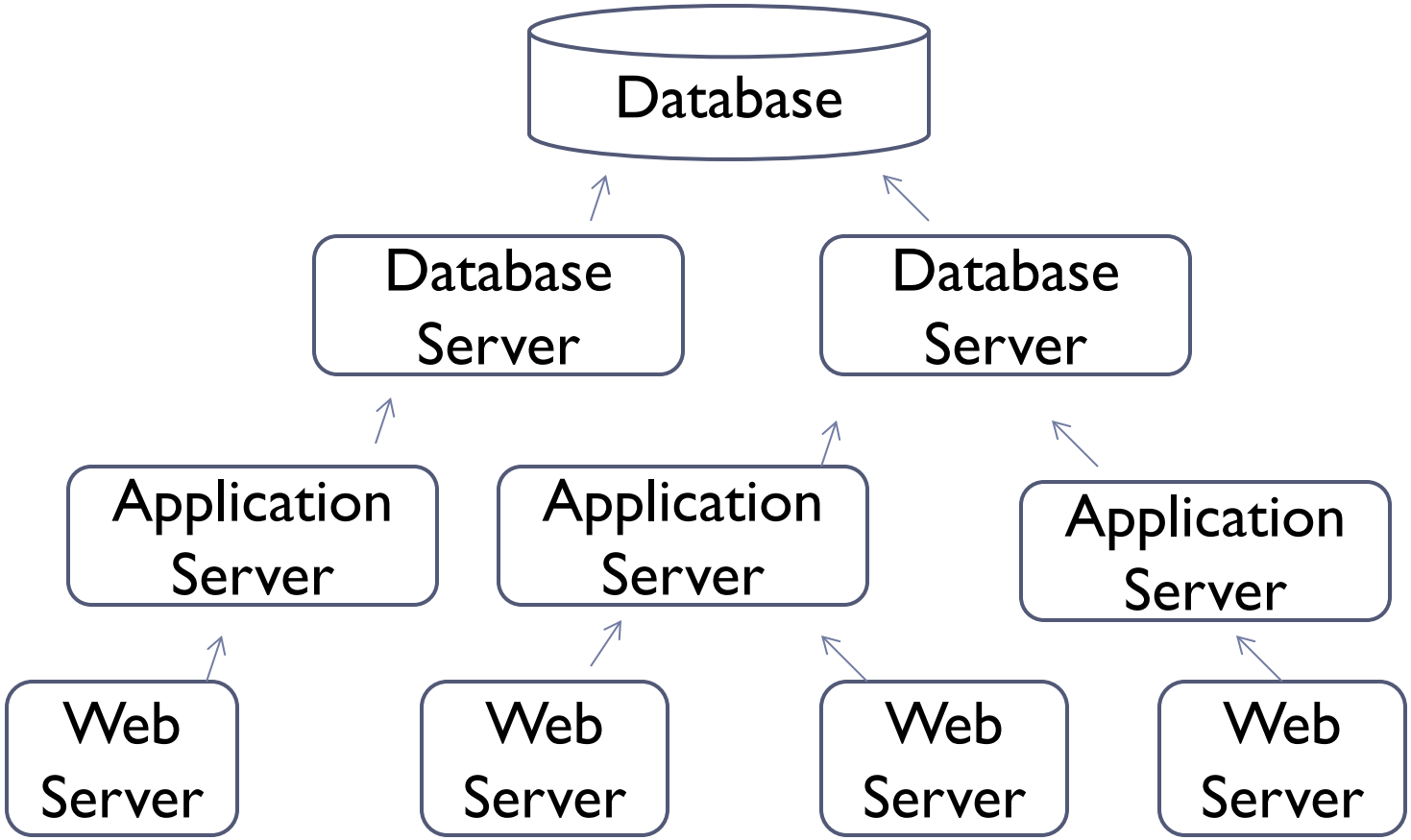
Database Programming

Three-Tier Architecture



SQL in Programs

Embedded SQL:

- SQL code blocks embedded in host language
- Additional commands (cursors, etc.)
- pre-processed into host-language code (fct/proc calls)

Dynamic SQL:

- Dynamic SQL code blocks in host language
- Queries are dynamic (Parameters, etc.)

Database Connectivity:

- CLI (Call-Level Interface):
- Dynamic SQL interface for programs
- ODBC, OLEDB, JDBC
- ORM (Object Relational Mapping)

Programming in SQL: SQL/PSM

PSM (Persistent Stored Module)

- stored in database (**stored procedure**)
- can be called from host-languages and SQL
- parameterized/programmed SQL

Vendors have proprietary versions of SQL/PSM

Oracle: **PL/SQL**

DB2: SQL/PL

SQL Server: Transact-SQL

What we'll do

Procedures and Functions

Procedure
- perform sequence of commands
- can include SQL, loops, conditionals, etc.
- can read through SQL statement one tuple at a time

Function
- like procedure but returns value

Function Example

```
create or replace function age_yr(year number)
       return number as
begin
  return extract(year from sysdate) - year;
end;
/

SELECT sid, age_yr(started)
FROM student;
```

Procedure Example

```
create or replace procedure enroll(sid number, cid
number, quarter varchar2, year number) as
begin
  INSERT INTO enrolled
  VALUES (sid, cid, quarter, year);
end;
/


call enroll(11035, 3201, 'Fall', 2015);
```

PL/SQL: variable declaration/assignment

```
declare
  val number := 1;
begin
  val := 1 + 2 * 3;
  dbms_output.put_line(val);
end;
/
```

- drop `declare` keyword for procedure/function bodies
- declared variables need not have default values assigned

PL/SQL: assignment

• can assign values of SQL statements that return a single value to variable using SELECT … INTO:

```
declare
  first_started number;
begin
  SELECT min(started) INTO first_started
  FROM student;
  dbms_output.put_line(first_started);
end;
/
```

• error message if SELECT returns no or multiple values or wrong type

PL/SQL: errors and exceptions

```
declare
  first_started number;
  sid number;
begin
  SELECT min(started) INTO first_started
  FROM student;
  SELECT SID INTO sid
  FROM student
  WHERE started = first_started;
  dbms_output.put_line(sid);
end;
```

- what if there are several students?

PL/SQL: errors and exceptions

```
declare
  first_started number;
  sid number;
begin
  SELECT min(started) INTO first_started
  FROM student;
  SELECT SID INTO sid
  FROM student
  WHERE started = first_started;
  dbms_output.put_line(sid);
exception
  when TOO_MANY_ROWS then
  dbms_output.put_line('Several students
in first year');
end;
```

PL/SQL: exceptions

```
DUP_VAL_ON_INDEX
NO_DATA_FOUND
TIMEOUT_ON_RESOURCE
TOO_MANY_ROWS
VALUE_ERROR
ZERO-DIVIDE


 WHEN OTHERS THEN
```

http://docs.oracle.com/cd/B10501_01/appdev.920/a96624/07_errs.htm

PL/SQL: variable declaration/assignment

```
create or replace function city_count(cname
varchar2) return number as
  cc number;
begin
  SELECT count(*) INTO cc
  FROM student
  WHERE city = cname;
  return cc;
end;
/

select distinct city, city_count('Chicago')
from student;
```

Simple Examples

- Write a procedure that deletes a student given by SID
- Write a procedure that deletes all students in a given year
- Given a course ID, a quarter and a year, calculate the number of students enrolled in the course at that time
- Given the name of a department, calculate the number of courses in the department
- For each student calculate how many courses they have enrolled in
- For each student calculate how many groups they are members of

PL/SQL: conditionals

```
set serveroutput on;
begin
  if dbms_random.value(0,1) > 0.5 then
      dbms_output.put_line('Head');
  else
      dbms_output.put_line('Tails');
  end if;
end;
/
```

```
if then end if;
if then else end if;
if then elsif then end if
```

More Examples

- Write a function that for each course returns whether it is ‘GRD’ or ‘UGRD’
- For every student compute their standing: freshman (< 3 courses), sophomore (< 5 courses), junior (< 7 courses), senior (everybody else).
- Given a student ID, determine whether the student enrolled during the current year (create output: dbms_output)
- (Requires prereq structure) When a student enrolls in a course, only allow this if we the student has already enrolled in all the prerequisite courses (use trigger)

PL/SQL: loops

```
set serveroutput on;
declare
 i number := 1;
begin
  loop
    i := i + 1;
    exit when i >= 10;
    dbms_output.put_line(i);
  end loop;
end;
/
```

Loop Examples

- Write code that computes the Fibonacci numbers (up to some bound)
- Create a look-up table for the Fibonacci numbers

PL/SQL: cursors

```
set serveroutput on;
declare
  cursor st_cursor IS
    (SELECT sid
     FROM student);
  st_id student.sid%type;
begin
  open st_cursor;
  loop
     fetch st_cursor INTO st_id;
     exit when st_cursor%notfound;
     dbms_output.put_line('Student ID: ' || st_id);
  end loop;
  close st_cursor;
end;
```

declare

attribute type

open

read

done

close

PL/SQL: cursors

```
set serveroutput on;
declare
  cursor st_cursor IS
    (SELECT sid, lastname, firstname
     FROM student);
  st_id student.sid%type;
  ln student.lastname%type;
  fn student.firstname%type;
begin
  open st_cursor;
  loop
     fetch st_cursor INTO st_id, ln, fn;
     exit when st_cursor%notfound;
     dbms_output.put_line('Student: ' || fn || ' ' || ln);
  end loop;
  close st_cursor;
end;
```

Cursor Examples

• Write a procedure that takes as input a course and department name and writes out the last year the course was offered (or a message that it has never been offered)
• Write a procedure that takes as input a course ID, cancels the course and sends a message “Dear *FirstName LastName*, your course *Department CourseName* has been cancelled” (can this be done in SQL?)
• Write a procedure that checks all student enrollments and drops graduate student enrollments in undergraduate classes and writes a warning message (sends email)
• Write a procedure that finds courses with the same name in the same department and cross-lists them: that is, we only keep the course with the largest CourseNr, delete all the others, and re-enroll students into the consolidated course (can this be done in SQL?)

Unnecessary loops

```
update employee
set salary = salary * 1.1
where salary < 90000;
update employee
set salary = salary *0.9
where salary >= 90000;
```

doesn't work

unnecessary

```
declare
   cursor emp_cursor IS
     (SELECT emp_id, salary
      FROM employee);
   e employee.emp_id%type;
   s employee.salary%type;
begin
   open emp_cursor;
   loop
      fetch emp_cursor INTO e,s;
      exit when emp_cursor%notfound;
      if s < 90000 then
       update employee
       set salary = s*1.1
       where emp_id = e;
      else
       update employee
       set salary = s*0.9
       where emp_id = e;
      end if;
   end loop;
   close emp_cursor;
end;
```

solution:

```
update employee
set salary =
    case when salary < 90000
         then salary * 1.1
         else salary * 0.9
    end;
```

“The best performance improvement technique for cursors inside the database is not to use them.”

Joe Celko

```
declare
  cursor emp_cursor IS
    (SELECT emp_id, salary
     FROM employee);
  e employee.emp_id%type;
  s employee.salary%type;
begin
  open emp_cursor;
  loop
     fetch emp_cursor INTO e,s;
     exit when emp_cursor%notfound;
     if s < 90000 then
      update employee
      set salary = s*1.1
      where emp_id = e;
     else
      update employee
      set salary = s*0.9
      where emp_id = e;
     end if;
  end loop;
  close emp_cursor;
end;
```