Databanken III

Chamilo

Oracle

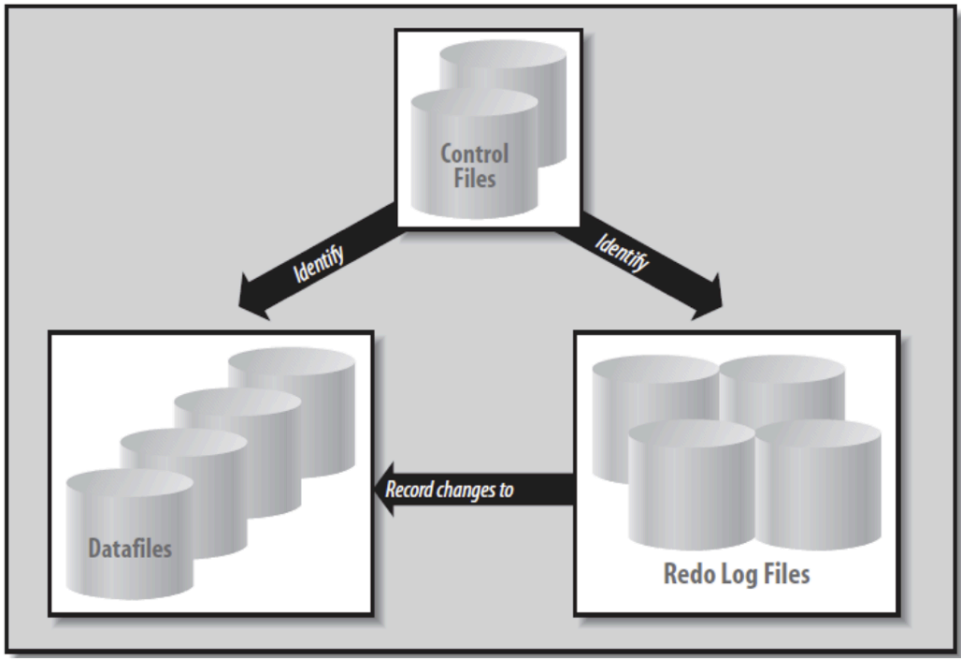
> **!** Oracle maakt geen gebruik Read Locks

- Oracle Instance [Logical]
- Oracle Database [Physical]

Files of a database

> Physical files

- Control files
- Datafiles
- Redo log files



Control files

- Database kan niet gestart worden zonder control files
- Binary file
- Bevat volgende informatie:
  - Naam van de database
  - Wanneer database gemaakt is
  - Naam en locatie van `datafiles` en `redo log files`
  - Character set dat gebruikt wordt om data op te slaan in de db
  - Checkpoint informatie
  - ...
- Minimum 2 control files op verschillende fysieke schijven

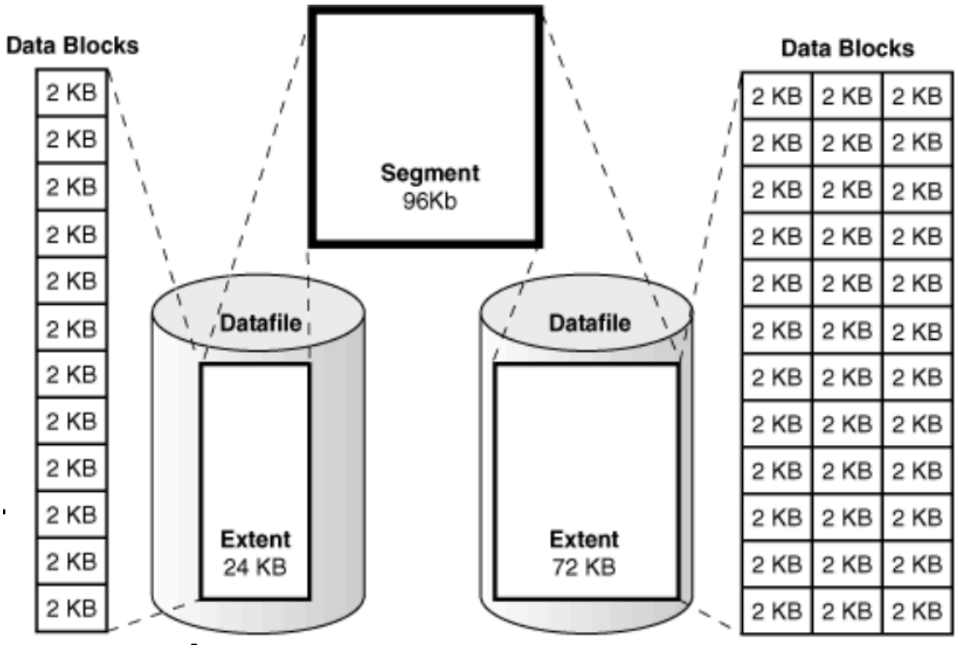
- Zonder kopie van control files -> risico op data verlies
- Locatie: gedefinieerd door `CONTROL_FILES` init parameter
- control_files = (/u00/oradata/control.001.dbf, /u01/oradata/control.002.dbf, /u02/oradata/control.003.dbf)

Datafiles

- Bevatten effectieve data
- bestaat uit database blocks
- behoren tot 1 database en 1 tablespace binnen de database
- wordt gelezen en beschreven in eenheden van oracle blocks van de datafiles naar memory
- datafile header
    - eerste block van elke datafile

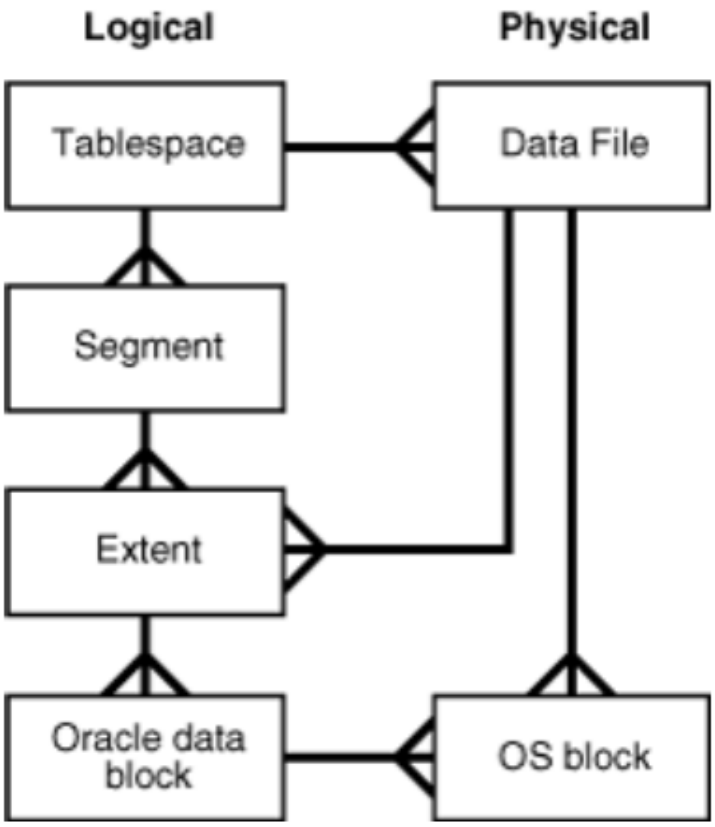
Data blocks, extents and segments

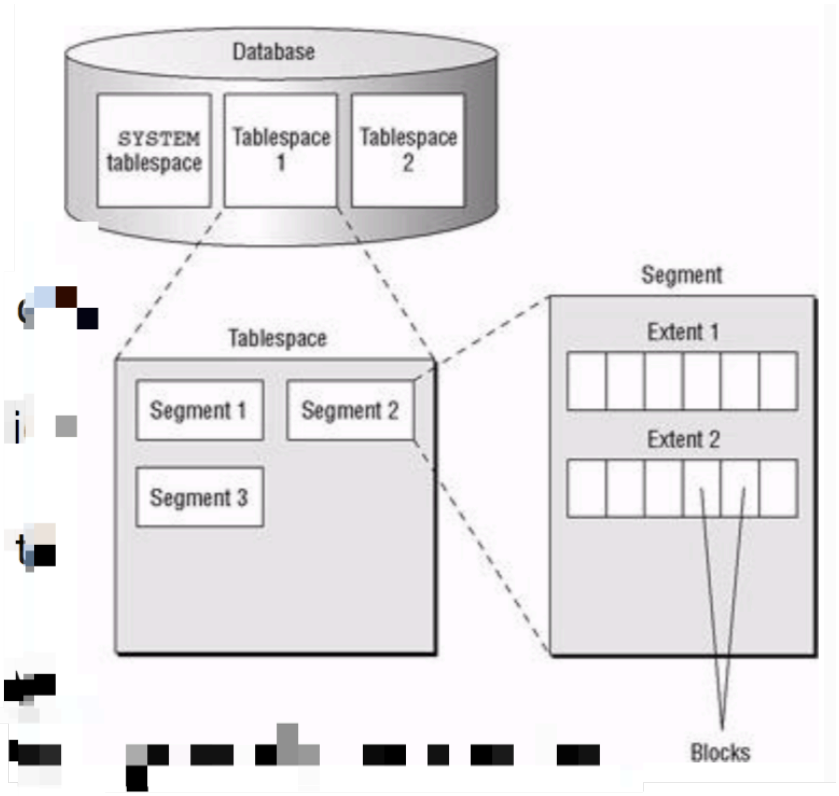
- fysiek standpunt: datafile = operating system blocks
- logisch standpunt: datafile = data blocks, extents and segments
- extent: groep van data blocks binnen een datafile
- segment: object gestored in de database, bestaat uit 1 of meer extents



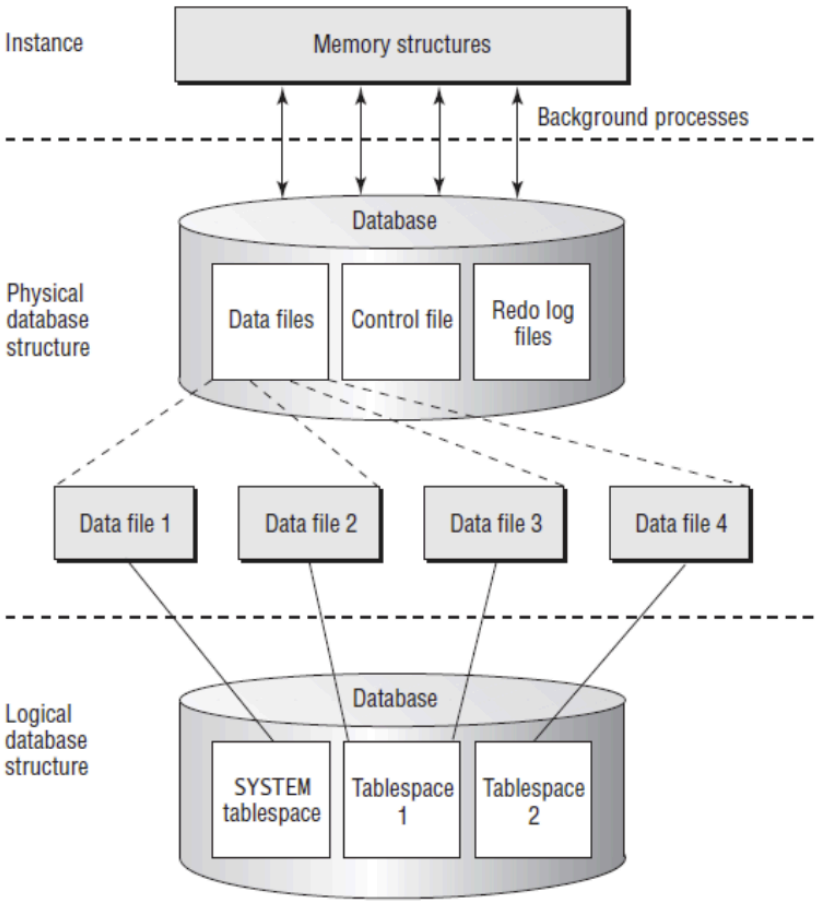
Tablespaces

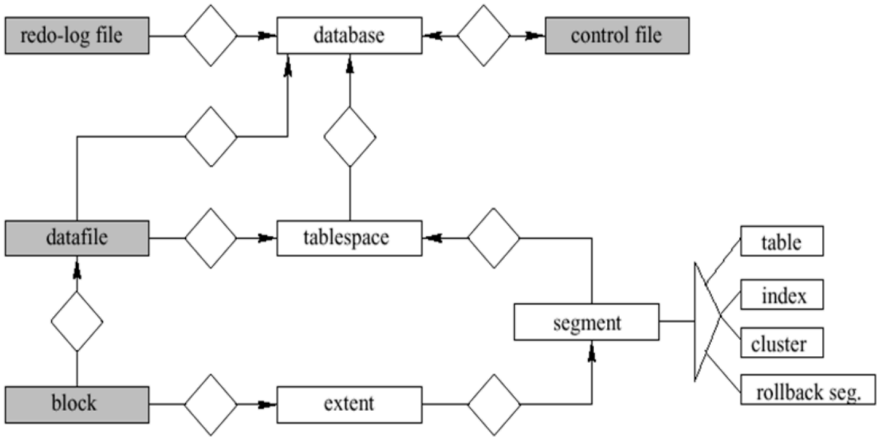
- bestaat uit 1 of meer datafiles
- elke datafile behoort tot 1 tablespace
- het is een logische structuur
- bij het maken van een tabel, kan je de tablespace specifiëren
- fundamentele eenheid van een oracle db object voor opslag, backup & recovery
- als er 1 datafile verloren gaat, is heel de tablespace onbeschikbaar
-

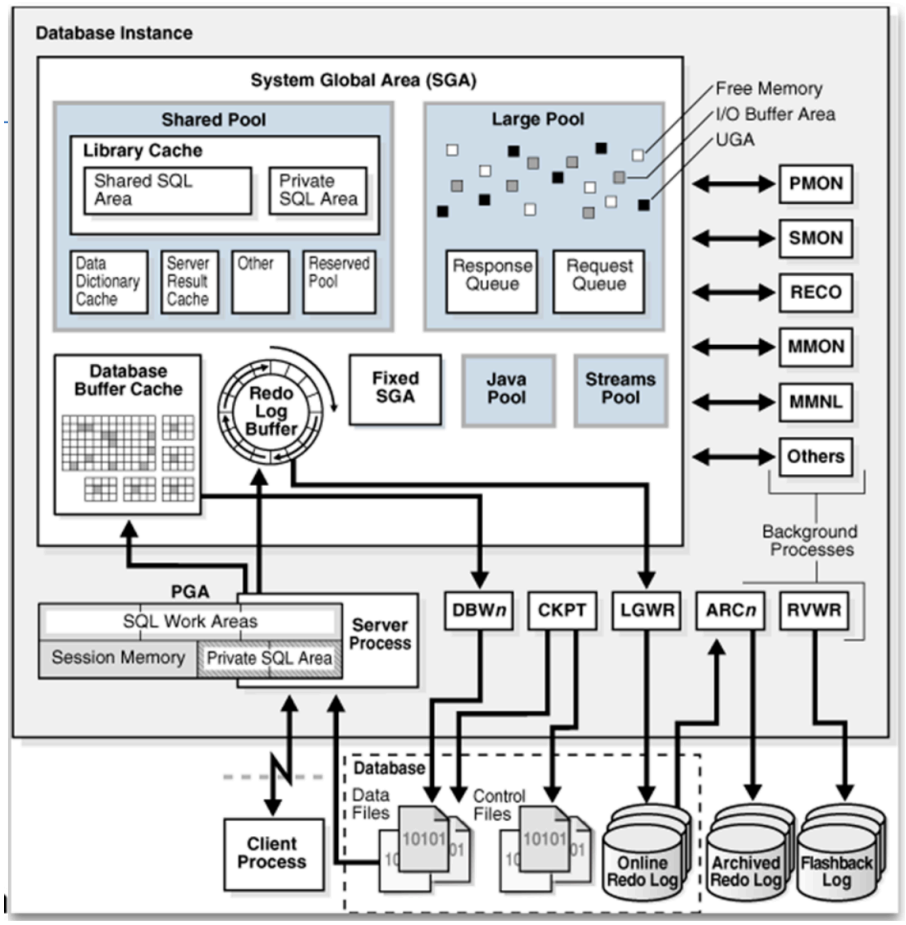




Fysieke structuur


Instance
Memory structures
Background processes
Database
Physical database structure
Data files
Control file
Redo log files
Data file 1
Data file 2
Data file 3
Data file 4
Database
Logical database structure
SYSTEM tablespace
Tablespace 1
Tablespace 2


redo-log file
database
control file
datafile
tablespace
table
index
segment
cluster
rollback seg.
block
extent



Hierarchy

1. Database
2. Tablespace
3. Segment
4. Extent
5. Block

PL/SQL

Slides

pdf	keywords
PLSQL_s01.pdf	introduction, benefits, creating PL/SQL blocks
PLSQL_s02.pdf	variables, types, %TYPE, functions, implicit/explicit data conversion, nested blocks, variable scope, <<outer>>, Good Programming Practices (¯_(ツ)_/¯), naming_conventions v_ (variable), c_ (constant), p_ (parameter)
PLSQL_s03.pdf	retrieving data, naming conventions, manipulating data, implicit cursors, transactional control statements
PLSQL_s04.pdf	Conditionals, IF, THEN, ELSE, ELSIF, CASE, basic loops, WHILE, FOR, nested loops, loop labels, <<outer_loop>>, <<inner_loop>>
PLSQL_s05.pdf	Explicit cursors, DECLARE, OPEN, FETCH, CLOSE, attributes, records, %ROWTYPE, %ISOPEN, %NOTFOUND, %FOUND, %ROWCOUNT, cursor FOR loops, cursor with parameters, cursors FOR UPDATE, NOWAIT, FOR UPDATE OF column-name, WHERE CURRENT OF cursor-name, multiple cursors
PLSQL_s06.pdf	User-Defined Records (custom types), Indexing Tables of Records
PLSQL_s07.pdf	Handling exceptions, EXCEPTION, WHEN, NO_DATA_FOUND, TOO_MANY_ROWS, OTHERS, 1. Predefined Oracle server error, 2. Non-predefiend Oracale server error, 3. User-defined error, INVALID_CURSOR, ZERO_DIVIDE, DUP_VAL_ON_INDEX, slide 32, SQLCODE, SQLERRM, slide 42, RAISE, RAISE_APPLICATION_ERROR, scope of exceptions
PLSQL_s08.pdf	creating PROCEDURE, PROCEDURE, CREATE OR REPLACE PROCEDURE xxx IS, parameters, DESCRIBE (slide 29), IN, OUT, IN OUT
PLSQL_s09.pdf	Creating FUNCTION, FUNCTION, CREATE OR REPLACE FUNCTION xxx IS, RETURN, USER, SYSDATE, IS, AS, Object Privilege (slide 38), AUTHID, CURRENT_USER
PLSQL_s10.pdf	creating PACKAGE, PACKAGE, CREATE OR REPLACE PACKAGE xxx IS, Package Body, CREATE OR REPLACE PACKAGE BODY xxx IS, DESCRIBE
PLSQL_s11.pdf	improving performance, NOCOPY, DETERMINISTIC, FORALL, BULK COLLECT, FETCH, RETURNING, SQL%BULK_ROWCOUNT(i), SAVE EXCEPTIONS, SQL%BULK_EXCEPTIONS.COUNT, SQL%BULK_EXCEPTIONS(i).ERROR_INDEX,

	`SQL%BULK_EXCEPTIONS(i).ERROR_CODE`
PLSQL_s12.pdf	`dynamic sql`, `EXEXECUTE IMMEDIATE xxx`, `INTO`, `USING`, `dynamic_string`, `define_variable`, `record`, `bind_argument`, `ALTER`

Exceptions

- **Oracle Predefined Exceptions:** oracle server errors, deze worden automatisch gegooid wanneer ze het willen.
  - Naam
  - Nummer
  - Foutboodschap
  - Worden door oracle zelf opgegooid
  - VB: NO_DATA_FOUND, TOO_MANY_ROWS (select into x, je mag dan maar 1 rij hebben)
- **Oracle Non-predefined Exceptions:**
  - ***geen*** Naam
  - wel nummer
  - wel foutboodschap
  - worden door oracle zelf opgegooid
  - VB: (nummer koppelen aan eigen exception): e_insert_excep EXCEPTION; PRAGMA EXCEPTION_INIT (e_insert_excep, -01400);
  - WHEN e_insert_excep THEN
- **User Defined Exceptions:** zelf definieren en opgooien (Slide 42)
  - Manier 1 (geen code of foutboodschap):
    1. Declareren: e_invalid_department EXCEPTION;
    2. Thrown: RAISE e_invalid_department;
    3. Catchen: WHEN e_invalid_department THEN
  1. Manier 2 (wel code & foutboodschap) code moet tussen -20000 en -30000:
     1. Declare: e_name EXCEPTION; PRAGMA EXCEPTION_INIT (e_name, -20999);
     2. Thrown: RAISE_APPLICATION_ERROR(-20999, 'Invalid last name');
     3. Catchen: WHEN e_name THEN

> ! `SQLCODE` + `SQLERRM` opvragen is mogelijk

```
/*
(1) Write a program that writes out the firstname, lastname and salary of
the CEO.
Catch any eexceptions (e.g. no CEO / multiple CEO's / ...)

See slide 28
*/

DECLARE
  v_first_name employees.firstname%TYPE;
  v_last_name employees.lastname%TYPE;
  v_salary employees.salary%TYPE;
BEGIN
  SELECT firstname, lastname, salary
  INTO v_first_name, v_last_name, v_salary
  FROM employees
  WHERE service = 'CEO';
  /*where service = 'CFO'  -- NO_DATA_FOUND*/
```

```
  /*where service = 'TRAINER' -- TOO_MANY_ROWS*/
  /*where service = 1 -- OTHERS*/

  DBMS_OUTPUT.PUT_LINE('CEO: ' || v_first_name || ' ' || v_last_name || '
has a salary of ' || v_salary);

  EXCEPTION
    WHEN TOO_MANY_ROWS THEN
      DBMS_OUTPUT.PUT_LINE('There are multiple CEOs');
    WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('There is no CEO');
    WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Another type of error occured with message: ' ||
sqlerrm || ' and code ' || sqlcode);
END;

/*
(2) Department id opgegeven.
Als department niet bestaat --> user defined exception
Anders: # werknemers voor department

Slide 52
*/

DECLARE
  v_department_id departments.departmentid%TYPE;

  e_non_existing_department EXCEPTION;
  PRAGMA EXCEPTION_INIT(e_non_existing_department, -20001);

  v_count_departments NUMBER(5, 0) := 0;
BEGIN
  v_department_id := '&Give_the_department_id';

  SELECT count(departmentid)
  INTO v_count_departments
  FROM departments
  WHERE departmentid = v_department_id;

  IF v_count_departments = 0 THEN
    RAISE_APPLICATION_ERROR(-20001, 'No such departments');
  END IF;

  EXCEPTION
    WHEN e_non_existing_department THEN
      DBMS_OUTPUT.PUT_LINE('SQLERRM: ' || SQLERRM || ' SQLCODE ' ||
SQLCODE);
END;
```

Procedures

```
/*
(1) Write a procedure that shows the data (firstname + lastname + service)
from all the employees on a given location.
Use a cursor to loop through the data. Throw a user defined exception if the
location doesn't exist
*/
```

```
CREATE OR REPLACE PROCEDURE showMeh(p_location varchar2) IS

 CURSOR employee_cursor IS
  SELECT firstname, lastname, service
  FROM employees
  INNER JOIN departments ON departments.departmentid = employees.department
  WHERE location = p_location;

  v_count number := 0;

  e_non_existing_location EXCEPTION;
  PRAGMA EXCEPTION_INIT(e_non_existing_location, -20001);

BEGIN

  SELECT count(location)
  INTO v_count
  FROM departments
  WHERE location = p_location;

  IF v_count = 0
  THEN
    RAISE_APPLICATION_ERROR(-20001, 'Location does not exist');
  END IF;

  FOR v_employee_record IN employee_cursor
  LOOP
    DBMS_OUTPUT.PUT_LINE(v_employee_record.firstname || ' ' ||
v_employee_record.lastname || ' - ' || v_employee_record.service);
  END LOOP;

  EXCEPTION
    WHEN e_non_existing_location THEN
      DBMS_OUTPUT.PUT_LINE('SQLERRM: ' || SQLERRM || ' SQLCODE ' ||
SQLCODE);
END;

/* TEST */
BEGIN
  DBMS_OUTPUT.PUT_LINE('TEST FOR BLABLA');
  DBMS_OUTPUT.PUT_LINE('');
  showMeh('BLABLA');

  DBMS_OUTPUT.PUT_LINE('');
  DBMS_OUTPUT.PUT_LINE('');

  DBMS_OUTPUT.PUT_LINE('TEST FOR ANTWERP');
  DBMS_OUTPUT.PUT_LINE('');
  showMeh('ANTWERP');
END;
```

Functions

```
/*
(1) The function checkService has a parameter p_service and returns true if
the service exists,
```

```
otherwise false + write a test program
*/

CREATE OR REPlACE FUNCTION checkService (p_service employees.service%TYPE)
RETURN boolean IS
  total number := 0;
BEGIN

  SELECT count(*)
  INTO total
  FROM employees
  WHERE service = p_service;

  return total <> 0;
END;

BEGIN
  IF checkService('TRAINER') THEN
    DBMS_OUTPUT.PUT_LINE('Service TRAINER exists');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Service TRAINER does not exists');
  END IF;
END;

/*
(2) The function checkSalary has a parameter p_service and p_salary and
returns true
if the salary is in a 15% range of the average salary for this service.
Otherwise the function returns false. + write a test program
*/

CREATE OR REPlACE FUNCTION checkSalary (p_service employees.service%TYPE,
p_salary employees.salary%TYPE)
RETURN boolean IS
  average number := 0;
  upperBound number := 0;
  lowerBound number := 0;
BEGIN

  SELECT avg(salary)
  INTO average
  FROM employees
  WHERE service = p_service;

  upperBound := average * 1.15;
  lowerBound := average - (average * .15);

  return (p_salary <= upperBound) AND (p_salary >= lowerBound);
END;

BEGIN
  IF checkSalary('TRAINER', 1970) THEN
    DBMS_OUTPUT.PUT_LINE('Salary is in a 15% range');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Salary is not in a 15% range');
  END IF;
END;
```

Package

```
CREATE OR REPLACE PACKAGE dbvideos_package1 IS

TYPE film_rec_type IS RECORD (
  bandcodee films.bandcode%TYPE,
  titel films.titel%TYPE,
  genre genres.genre%TYPE,
  prijs films.prijs%TYPE,
  aantalkeerverhuurd films.totverhuurd%TYPE
);

TYPE film_tab_type IS TABLE OF film_rec_type INDEX BY BINARY_INTEGER;

PROCEDURE films_per_genre;

FUNCTION geef_pop_films_per_leeftcat (p_min_leeftijd NUMBER, p_max_leeftijd
NUMBER, p_aantal NUMBER) RETURN film_tab_type;

FUNCTION wijzig_prijs (p_percentage NUMBER, p_maatschappij
maatschappijen.maatschappijnaam%TYPE) RETURN NUMBER;

e_video_exception EXCEPTION;
PRAGMA EXCEPTION_INIT(e_video_exception, -20001);

END dbvideos_package1;



CREATE OR REPLACE PACKAGE BODY dbvideos_package1 IS

PROCEDURE films_per_genre IS
CURSOR c_genres IS SELECT * FROM genres;
CURSOR c_films (p_genrecode films.genrecode%TYPE) IS SELECT bandcode, titel
FROM films WHERE genrecode = p_genrecode;
BEGIN
  FOR v_genrerecord IN c_genres LOOP
    DBMS_OUTPUT.PUT_LINE('Genre ' || v_genrerecord.genre);

    FOR v_filmrec IN c_films (v_genrerecord.genrecode) LOOP
      DBMS_OUTPUT.PUT_LINE(v_filmrec.bandcode || ' ' || v_filmrec.titel);
    END LOOP;
  END LOOP;
END;

FUNCTION geef_pop_films_per_leeftcat (p_min_leeftijd NUMBER, p_max_leeftijd
NUMBER, p_aantal NUMBER) RETURN film_tab_type IS
  v_result film_tab_type;
BEGIN
  SELECT f.bandcode, f.titel, g.genre, f.prijs, count(v.bandcode)
  BULK COLLECT INTO v_result
  FROM films f JOIN genres g ON f.genrecode = g.genrecode
  JOIN verhuur v ON v.bandcode = f.bandcode
  JOIN klantenvideo k ON k.klantcode = v.klantcode
  WHERE TO_NUMBER(TO_CHAR(SYSDATE, 'YYYY')) -
TO_NUMBER(TO_CHAR(k.geboortedatum, 'YYYY')) BETWEEN p_min_leeftijd AND
p_max_leeftijd
  AND ROWNUM <= p_aantal
  GROUP BY f.bandcode, f.titel, g.genre, f.prijs
  ORDER BY count(v.bandcode) DESC;

  RETURN v_result;
```

```
END;

FUNCTION wijzig_prijs (p_percentage NUMBER, p_maatschappij
maatschappijen.maatschappijnaam%TYPE) RETURN NUMBER IS
  v_count NUMBER;
  TYPE bandcodes_tab_type IS TABLE OF films.bandcode%TYPE INDEX BY
BINARY_INTEGER;
  v_bandcodes bandcodes_tab_type;
BEGIN
  SELECT count(maatschappijcode) INTO v_count
  FROM maatschappijen
  WHERE maatschappijnaam = p_maatschappij;
  IF v_count = 0 THEN
    RAISE_APPLICATION_ERROR(-20001, 'De maatschappij bestaat niet');
  END IF;
  SELECT bandcode BULK COLLECT INTO v_bandcodes
  FROM films f JOIN maatschappijen m
  ON f.maatschappijcode = m.maatschappijcode
  WHERE m.maatschappijnaam = p_maatschappij;

  FORALL i IN v_bandcodes.FIRST .. v_bandcodes.LAST
    UPDATE films
    SET prijs = prijs * (1 + p_percentage)
    WHERE bandcode = v_bandcodes(i);

  EXCEPTION
    WHEN e_video_exception THEN
      DBMS_OUTPUT.PUT_LINE('SQLCODE ' || ' ' || SQLCODE);
    WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('SQLCODE ' || ' ' || SQLCODE);
END;

END dbvideos_package1;
```

Afsluitende herhalings oefeningen

```
-- Oefening 1

-- Declaratie van de package specification
CREATE OR REPLACE PACKAGE dbvideos_package1 IS
    TYPE film_rec_type IS RECORD (
      bandcodee films.bandcode%TYPE,
      titel films.titel%TYPE,
      genre genres.genre%TYPE,
      prijs films.prijs%TYPE,
      aantalkeerverhuurd films.totverhuurd%TYPE
    );

    TYPE film_tab_type IS TABLE OF film_rec_type INDEX BY BINARY_INTEGER;

    PROCEDURE films_per_genre;

    FUNCTION geef_pop_films_per_leeftcat (p_min_leeftijd NUMBER,
p_max_leeftijd NUMBER, p_aantal NUMBER) RETURN film_tab_type;

    FUNCTION wijzig_prijs (p_percentage NUMBER, p_maatschappij
maatschappijen.maatschappijnaam%TYPE) RETURN NUMBER;
```

```
    e_video_exception EXCEPTION;
    PRAGMA EXCEPTION_INIT(e_video_exception, -20001);
END dbvideos_package1;

-- Declaratie van de package body
CREATE OR REPLACE PACKAGE BODY dbvideos_package1 IS

    procedure films_per_genre IS
        CURSOR c_genres IS SELECT * FROM genres;
        CURSOR c_films (p_genrecode genres.genrecode%TYPE) IS
        SELECT bandcode, titel FROM films WHERE genrecode = p_genrecode;
        BEGIN
            FOR v_genrerec IN c_genres LOOP
                DBMS_OUTPUT.PUT_LINE('Genre ' || v_genrerec.genre);
                FOR v_filmrec IN c_films (v_genrerec.genrecode)
                LOOP
                DBMS_OUTPUT.PUT_LINE(v_filmrec.bandcode ||
                    ' ' || v_filmrec.titel);
                END LOOP;
            END LOOP;
        END;

function geef_pop_films_per_leeftcat (p_min_leeftijd NUMBER,
        p_max_leeftijd NUMBER, p_aantal NUMBER) RETURN film_tab_type IS
    v_result film_tab_type;

BEGIN
    SELECT f.bandcode, f.titel, g.genre, f.prijs, COUNT(v.bandcode)
    BULK COLLECT INTO v_result
    FROM films f JOIN genres g ON f.genrecode = g.genrecode
    JOIN verhuur v ON v.bandcode = f.bandcode
    JOIN klantenvideo k ON v.klantcode = k.klantcode
    WHERE TO_NUMBER(TO_CHAR(SYSDATE,'YYYY')) -
        TO_NUMBER(TO_CHAR(k.geboortedatum, 'YYYY'))
        BETWEEN p_min_leeftijd AND p_max_leeftijd
    AND WHERE rownum <= p_aantal
    GROUP BY f.bandcode, f.titel, g.genre, f.prijs
    ORDER BY COUNT(v.bandcode) DESC;

    RETURN v_result;
END;

function wijzig_prijs (p_percentage NUMBER, p_maatschappij
    maatschappijen.maatschappijnaam%TYPE) RETURN NUMBER IS
    TYPE bandcodes_tab_type IS TABLE OF films.bandcode%TYPE
    INDEX BY BINARY_INTEGER;
    v_bandcodes_tab bandcodes_tab_type;
    v_count NUMBER;

BEGIN
    SELECT COUNT(maatschappijcode) INTO v_count
    FROM maatschappijen
    WHERE maatschappijnaam = p_maatschappij;
    IF v_count = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Maatschappij doesn''t exist');
    END IF;
    SELECT f.bandcode BULK COLLECT INTO v_bandcodes_tab
    FROM films f JOIN maatschappijen m
    ON f.maatschappijcode = m.maatschappijcode
    WHERE m.maatschappijnaam = p_maatschappij;
```

```
    FORALL i in v_bandcodes_tab.FIRST .. v_bandcodes_tab.LAST
        UPDATE films
        SET prijs = prijs * (1 + p_percentage)
        WHERE bandcode = v_bandcodes_tab(i);

    RETURN v_bandcodes_tab.COUNT();
END;

END dbvideos_package1;

/* Testcode */
DECLARE
    v_film_tab  dbvideos_package1.film_tab_type;
    v_count NUMBER;

BEGIN
    dbvideos_package1.films_per_genre();
    v_film_tab := dbvideos_package1.geef_pop_films_per_leeftcat(20, 25,5);
    FOR i in v_film_tab.FIRST .. v_film_tab.LAST LOOP
        IF v_film_tab.EXISTS(i) THEN
            DBMS_OUTPUT.PUT_LINE(v_film_tab(i).titel || ' '
            || v_film_tab(i).genre);
        END IF;
    END LOOP;
    v_count := dbvideos_package1.wijzig_prijs(0.05, 'HOLLYWOOD VIDEO');
    DBMS_OUTPUT.PUT_LINE('aantal gewijzigde records is ' || v_count);
END;

/**********
Oefening 2
**********/

/* Declaratie van de package specification */
create or replace package dbvideos_package2  IS
TYPE genre_rec_type IS RECORD (genrecode genres.genrecode%TYPE, genre
genres.genre%TYPE,
                               aantal INTEGER);
TYPE genre_tab_type is TABLE OF genre_rec_type INDEX BY PLS_INTEGER;
  e_video_exception EXCEPTION;
  PRAGMA EXCEPTION_INIT(e_video_exception, -20001);

procedure films_per_datum;
function geef_genres_per_klant (p_klantcode klantenvideo.klantcode%TYPE)
return genre_tab_type;
function wijzig_aantal_exemplaren(p_maximum NUMBER) return NUMBER;
END VideoPackage;
create or replace package BODY dbvideos_package2 IS
procedure films_per_datum as
cursor cur_datum is select distinct verhuurdatum from verhuur order by
verhuurdatum;
cursor cur_films (p_verhuurdatum verhuur.verhuurdatum%type) is
   select distinct f.bandcode, f.titel, g.genre from films f join genres g
   on g.genrecode = f.genrecode join verhuur v on f.bandcode = v.bandcode
   where v.verhuurdatum = p_verhuurdatum
   order by f.titel;
begin
  for datum_rec in cur_datum loop
    dbms_output.put_line('Datum ' || datum_rec.verhuurdatum);
    for film_rec in cur_films(datum_rec.verhuurdatum) loop
      dbms_output.put_line(film_rec.bandcode || ' ' || film_rec.titel);
```

```
    end loop;
      dbms_output.put_line('');
  end loop;
END;

function geef_genres_per_klant (p_klantcode klantenvideo.klantcode%TYPE)
return genre_tab_type IS
v_resultaat genre_tab_type;
  v_aantal NUMBER(6,0);
  v_teller NUMBER(6,0);
cursor cur_genres is
   select distinct g.genrecode, g.genre, count(v.bandcode) As aantal from
films f join genres g
   on g.genrecode = f.genrecode join verhuur v on f.bandcode = v.bandcode
   where v.klantcode = p_klantcode
   group by g.genrecode, g.genre;
BEGIN
  SELECT count(klantcode) into v_aantal from klantenvideo where klantcode =
p_klantcode;
  if v_aantal = 0 then
      RAISE_APPLICATION_ERROR(-20001, 'Er is iets fout met de
maatschappijnaam');
  end if;

  v_teller := 1;
  for genre_rec in cur_genres loop
    v_resultaat(v_teller).genrecode := genre_rec.genrecode;
    v_resultaat(v_teller).genre := genre_rec.genre;
    v_resultaat(v_teller).aantal := genre_rec.aantal;
    v_teller := v_teller + 1;
  end loop;

  return v_resultaat;
EXCEPTION
WHEN  e_video_exception then
dbms_output.put_line (SQLERRM);
WHEN OTHERS THEN
dbms_output.put_line ('Er is een fout opgetreden!');

END;

function wijzig_aantal_exemplaren(p_maximum NUMBER) return NUMBER IS
TYPE t_films IS TABLE OF films.bandcode%TYPE INDEX BY BINARY_INTEGER;
  v_films_tab t_films;

BEGIN
  SELECT bandcode BULK COLLECT INTO v_films_tab FROM films
  WHERE voorraad + verhuurd < p_maximum;

  FORALL i IN v_films_tab.FIRST..v_films_tab.LAST
  UPDATE films
  SET voorraad = p_maximum - verhuurd
  WHERE bandcode = v_films_tab(i);

  return v_films_tab.count();
  END;

end videopackage;

/* Testprogramma */
```

```
DECLARE
v_aantal NUMBER(6,0) := 0;
v_resultaat videopackage.genre_tab_type;
begin
videopackage.films_per_datum();
v_resultaat := videopackage.geef_genres_per_klant(6);
  FOR i IN v_resultaat.FIRST..v_resultaat.LAST LOOP
      IF v_resultaat.EXISTS(i) THEN
    DBMS_OUTPUT.PUT_LINE(v_resultaat(i).genrecode || ' ' ||
v_resultaat(i).genre || ' - ' || v_resultaat(i).aantal);
  END IF;
  END LOOP;
  dbms_output.put_line('');
  v_aantal := videopackage.wijzig_aantal_exemplaren(5);
  dbms_output.put_line('aantal gewijzigde records is ' || v_aantal);

end;
```

Hadoop

1. `start-dfs.sh`
2. `hadoop fs -ls`
3. `hadoop fs -mkdir input`
4. `hadoop fs -ls input`
5. `hadoop fs -put someFile input` om someFile in input te stoppen
6. `hadoop -put * input` om alles in input te stoppen
7. `cd workspace/quickstart/`
8. `spark-submit --class bdstudents.quickstart.App --master local[2] target/quickstart-1.0.jar`

Spark

Setup

```
SparkConf conf = new SparkConf().setAppName("Simple Application");
JavaSparkContext sc = new JavaSparkContext(conf);
```

Compilen

Met 4 Maven Build

Alle logs afzetten

```
// ...
import org.apache.log4j.Level;
import org.apache.log4j.Logger;

public class App {
    public static void main(String[] args) {
        Logger.getLogger("org").setLevel(Level.OFF);
        Logger.getLogger("akka").setLevel(Level.OFF);

        // ...
```

```
    }
}
```

Les 1

```
JavaRDD<String> inputRDD = sc.textFile("/user/hduser/input/test");
JavaRDD<String> johnRDD = inputRDD.filter(line -> line.contains("john"));
JavaRDD<String> janeRDD = inputRDD.filter(line -> line.contains("jane"));
JavaRDD<String> resultRDD = sc.emptyRDD();

resultRDD = johnRDD.union(janeRDD);
System.out.println("Aantal keren John en Jane" + resultRDD.count());

System.out.println(resultRDD.collect());
sc.close();
```

Les 2

```
JavaPairRDD<String, StockPriceDate> resultRDD =
sc.textFile("/user/hduser/input/NYSE-2000-2001.tsv")
    .filter(line -> line.contains("ASP"))
    .map(line -> line.split("\t"))
    .mapToPair(fields -> new Tuple2<String, StockPriceDate>(fields[1], new
StockPriceDate(Double.parseDouble(fields[6]), fields[2])))
    .reduceByKey((x, y) -> x.getStockprice() > y.getStockprice() ? x : y)
    .sortByKey();

System.out.println(resultRDD.collect());
sc.close();
```

Les 3

```
// 3.1
// ---

JavaPairRDD<String, Integer> resultRDD =
sc.textFile("/user/hduser/input/graph_data_part1.txt")
        .map(line -> line.split("\t"))
        .mapToPair(fields -> new Tuple2<String, Integer>(fields[1] + "_" +
fields[2], 1))
        .reduceByKey((x, y) -> x + y)
        .filter(line -> line._2 == 3);

// 3.2
// ---
// OPTIE 1:
JavaPairRDD<String, Integer> resultRDD =
sc.textFile("/user/hduser/input/graph_data_part2.txt")
        .filter(line -> line.length() > 0)
        .distinct() // dure operatie want: bestand is verdeeld in blokken op
de server. Per lijn gaat hij alle servers contacteren om te zien of hij
```

```
unique is.
        .map(line -> line.split("\t"))
        .mapToPair(fields -> new Tuple2<String, Integer>(fields[1] + "_" +
fields[2], 1))
        .reduceByKey((x, y) -> x + y)
        .filter(line -> line._2 == 3);

// OPTIE 2:
JavaPairRDD<String, Integer> resultRDD =
sc.textFile("/user/hduser/input/graph_data_part2.txt")
        .filter(line -> line.length() > 0)
        .map(line -> line.split("\t"))
        .mapToPair(fields -> new Tuple2<String, String>(fields[1] + "_" +
fields[2], fields[0]))
        .combineByKey(
            x -> new HashSet<String>(Arrays.asList(x)),
            (x, y) -> { x.add(y); return x; },
            (x, y) -> { x.addAll(y); return x; }
        )
        .mapValues(arr -> arr.size())
        .filter(line -> line._2 == 3);

// 3.6
// ---

List<String> files = sc.wholeTextFiles("/user/hduser/input/symptomen")
        .keys()
        .collect();

JavaRDD<String> emptyRDD = sc.emptyRDD();
JavaPairRDD<String, String> resultRDD = emptyRDD
        .mapToPair(fields -> new Tuple2<String, String>("", ""));

for (int i = 0; i < files.size(); i++) {
    String path = files.get(i);

    int positionLastSlash = path.lastIndexOf("/");
    String disease = path.substring(positionLastSlash + 1);

    JavaPairRDD<String, String> extraRDD = sc.textFile(path)
            .filter(line -> line.length() > 0)
            .map(line -> line.split("\t"))
            .mapToPair(fields -> new Tuple2<String, String>(disease,
fields[0]));
    resultRDD = resultRDD.union(extraRDD);
}

JavaPairRDD<String, HashSet<String>> extraRDD = resultRDD
        .mapToPair(t -> new Tuple2<String, String>(t._2, t._1))
        .combineByKey(
            x -> new HashSet<String>(Arrays.asList(x)),
            (x, y) -> { x.add(y); return x; },
            (x, y) -> { x.addAll(y); return x; }
        )
        .filter(t -> t._2.size() == 1);
```

Used Data sets

- /user/hduser/input/names

```
jan
an
stijn
els
an
els
steven
```

- /user/hduser/input/scores

```
jan     5
jan     9
stijn   3
jan     7
jan     8
stijn   6
```

- /user/hduser/input/data

```
an      20  V   9000
jan     25  M   9100
stijn   30  M   9000
tine    24  V   8500
```

Parallelize

```
JavaRDD<Integer> rdd = sc.parallelize(Arrays.asList(1, 2, 3));
```

textFile

```
JavaRDD<String> inputRDD = sc.textFile("/user/hduser/input/log");
```

wholeTextFiles

```
JavaRDD<String> inputRDD = sc.wholeTextFiles("/user/hduser/input/");
```

map

```
package bdstudents.quickstart;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.*;
```

```
public class App {
    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setAppName("Simple Application");
        JavaSparkContext sc = new JavaSparkContext(conf);

        JavaRDD<String> resultRDD = sc.textFile("/user/hduser/input/scores")
                                      .map(line -> line.toUpperCase());

        System.out.println(resultRDD.collect());
        sc.close();
    }
}
```

> The result is [JAN 5, JAN 9, STIJN 3, JAN 7, JAN 8, STIJN 6]

```
package bdstudents.quickstart;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.*;
import java.util.Arrays;

public class App {
    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setAppName("Simple Application");
        JavaSparkContext sc = new JavaSparkContext(conf);

        JavaRDD<String []> resultRDD =
sc.textFile("/user/hduser/input/scores")
                                        .map(line -> line.split("\t"));
        resultRDD.foreach(t -> System.out.println(Arrays.toString(t)));

        sc.close();
    }
}
```

> The result is
>
> [jan, 5]
> [jan, 8]
> [stijn, 6]
> [jan, 9]
> [stijn, 3]
> [jan, 7]

filter

```
package bdstudents.quickstart;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.*;

public class App {
    public static void main(String[] args) {
```

```
        SparkConf conf = new SparkConf().setAppName("Simple Application");
        JavaSparkContext sc = new JavaSparkContext(conf);

        JavaRDD<String> resultRDD = sc.textFile("/user/hduser/input/scores")
                                      .filter(line -> line.length() > 0)
                                      .map(line -> line.toUpperCase());

        System.out.println(resultRDD.collect());
        sc.close();
    }
}
```

> The result is [JAN 5, JAN 9, STIJN 3, JAN 7, JAN 8, STIJN 6]

flatMap

```
package bdstudents.quickstart;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.*;
import java.util.Arrays;

public class App {
    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setAppName("Simple Application");
        JavaSparkContext sc = new JavaSparkContext(conf);

        JavaRDD<String> resultRDD = sc.textFile("/user/hduser/input/scores")
            .flatMap(line -> Arrays.asList(line.split("\t")).iterator());

        System.out.println(resultRDD.collect());
        sc.close();
    }
}
```

> The result is [jan, 5, jan, 9, stijn, 3, jan, 7, jan, 8, stijn, 6]

union

```
package bdstudents.quickstart;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.*;

public class App {
    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setAppName("Simple Application");
        JavaSparkContext sc = new JavaSparkContext(conf);

        JavaRDD<String> inputRDD = sc.textFile("/user/hduser/input/names");
        JavaRDD<String> anRDD = inputRDD.filter(line ->
line.contains("an"));
        JavaRDD<String> elsRDD = inputRDD.filter(line ->
line.contains("els"));
        JavaRDD<String> resultRDD = anRDD.union(elsRDD);
```

```
        // Equivalent:
        JavaRDD<String> resultRDD = sc.textFile("/user/hduser/input/names")
        .filter(line -> line.contains("an") || line.contains("els"));

        System.out.println(resultRDD.collect());
        sc.close();
    }
}
```

> The result is [jan, an, an, els, els]

intersection

```
package bdstudents.quickstart;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.*;

public class App {
    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setAppName("Simple Application");
        JavaSparkContext sc = new JavaSparkContext(conf);

        JavaRDD<String> inputRDD = sc.textFile("/user/hduser/input/names");
        JavaRDD<String> janRDD = inputRDD.filter(line ->
line.contains("jan"));
        JavaRDD<String> anRDD = inputRDD.filter(line ->
line.contains("an"));
        JavaRDD<String> resultRDD = anRDD.intersection(janRDD);

        System.out.println(resultRDD.collect());
        sc.close();
    }
}
```

> The result is [jan]

subtract

```
package bdstudents.quickstart;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.*;

public class App {
    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setAppName("Simple Application");
        JavaSparkContext sc = new JavaSparkContext(conf);

        JavaRDD<String> inputRDD = sc.textFile("/user/hduser/input/names");
        JavaRDD<String> janRDD = inputRDD.filter(line ->
line.contains("jan"));
        JavaRDD<String> anRDD = inputRDD.filter(line ->
line.contains("an"));
        JavaRDD<String> resultRDD = anRDD.subtract(janRDD);
```

```
        System.out.println(resultRDD.collect());
        sc.close();
    }
}
```

> The result is [an, an]

distinct

```
package bdstudents.quickstart;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.*;

public class App {
    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setAppName("Simple Application");
        JavaSparkContext sc = new JavaSparkContext(conf);

        JavaRDD<String> resultRDD = sc.textFile("/user/hduser/input/names")
        .filter(line -> line.contains("an"))
        .distinct();

        System.out.println(resultRDD.collect());
        sc.close();
    }
}
```

> The result is [jan, an]

groupBy

```
package bdstudents.quickstart;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.*;

public class App {
    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setAppName("Simple Application");
        JavaSparkContext sc = new JavaSparkContext(conf);

        JavaPairRDD<String, Iterable<String>> resultRDD =
        sc.textFile("/user/hduser/input/data")
        .groupBy(line -> Arrays.asList(line.split("\t")).get(3));

        System.out.println(resultRDD.collect());
        sc.close();
    }
}
```

> The result is [(9100,[jan 25 M 9100]), (8500,[tine 24 V 8500]), (9000,[an 20 V 9000, stijn 30 M 9000])]

keyBy

```
package bdstudents.quickstart;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.*;

public class App {
    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setAppName("Simple Application");
        JavaSparkContext sc = new JavaSparkContext(conf);

        JavaPairRDD<String, String> resultRDD =
        sc.textFile("/user/hduser/input/data")
        .keyBy(line -> Arrays.asList(line.split("\t")).get(3));

        System.out.println(resultRDD.collect());
        sc.close();
    }
}
```

> The result is [(9000,an 20 V 9000), (9100,jan 25 M 9100), (9000,stijn 30 M 9000), (8500,tine 24 V 8500)]

sample

```
package bdstudents.quickstart;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.*;

public class App {
    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setAppName("Simple Application");
        JavaSparkContext sc = new JavaSparkContext(conf);

        JavaRDD<String> resultRDD = sc.textFile("/user/hduser/input/data")
        .sample(true, 0.5);

        System.out.println(resultRDD.collect());
        sc.close();
    }
}
```

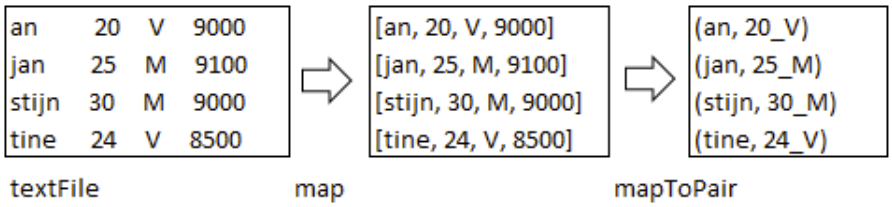
> The result is [an 20 V 9000]

mapToPair

```
package bdstudents.quickstart;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.*;
import scala.Tuple2;
```

```
public class App {
    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setAppName("Simple Application");
        JavaSparkContext sc = new JavaSparkContext(conf);

        JavaPairRDD<String, String> resultRDD =
        sc.textFile("/user/hduser/input/data")
        .map(line -> line.split("\t"))
        .mapToPair(fields -> new Tuple2<String, String>(fields[0], fields[1]
+ "_" + fields[2]));

        System.out.println(resultRDD.collect());
        sc.close();
    }
}
```

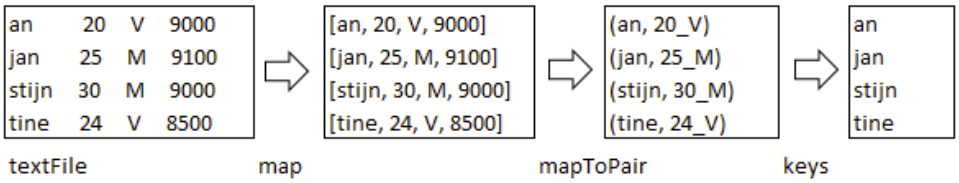
> The result is [(an,20_V), (jan,25_M), (stijn,30_M), (tine,24_V)]



keys

```
package bdstudents.quickstart;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.*;
import scala.Tuple2;

public class App {
    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setAppName("Simple Application");
        JavaSparkContext sc = new JavaSparkContext(conf);

        JavaPairRDD<String, String> resultRDD =
        sc.textFile("/user/hduser/input/data")
        .map(line -> line.split("\t"))
        .mapToPair(fields -> new Tuple2<String, String>(fields[0], fields[1]
+ "_" + fields[2]))
        .keys();

        // Equivalent

        JavaRDD<String> resultRDD = sc.textFile("/user/hduser/input/data")
        .map(line -> line.split("\t")[0]);

        System.out.println(resultRDD.collect());
        sc.close();
    }
}
```

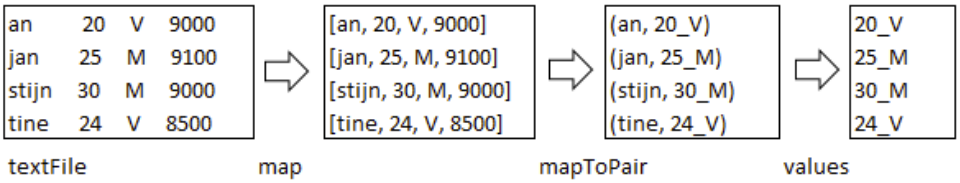
> The result is [an, jan, stijn, tine]



values

```
package bdstudents.quickstart;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.*;
import scala.Tuple2;

public class App {
    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setAppName("Simple Application");
        JavaSparkContext sc = new JavaSparkContext(conf);

        JavaPairRDD<String, String> resultRDD =
        sc.textFile("/user/hduser/input/data")
        .map(line -> line.split("\t"))
        .mapToPair(fields -> new Tuple2<String, String>(fields[0], fields[1]
+ "_" + fields[2]))
        .values();

        System.out.println(resultRDD.collect());
        sc.close();
    }
}
```

> The result is [20_V, 25_M, 30_M, 24_V]

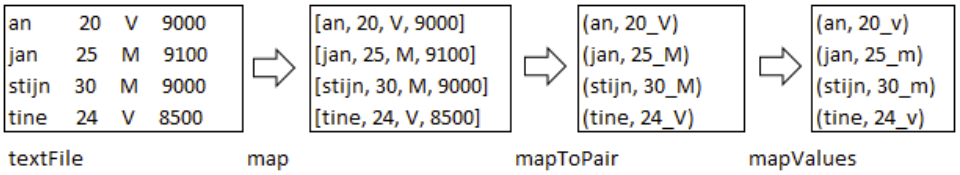


mapValues

```
package bdstudents.quickstart;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.*;
import scala.Tuple2;
public class App {
    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setAppName("Simple Application");
        JavaSparkContext sc = new JavaSparkContext(conf);

        JavaPairRDD<String, String> resultRDD =
        sc.textFile("/user/hduser/input/data")
        .map(line -> line.split("\t"))
        .mapToPair(fields -> new Tuple2<String, String>(fields[0], fields[1]
+ "_" + fields[2]))
```

```
        .mapValues(line -> line.toLowerCase());

        System.out.println(resultRDD.collect());
        sc.close();
    }
}
```

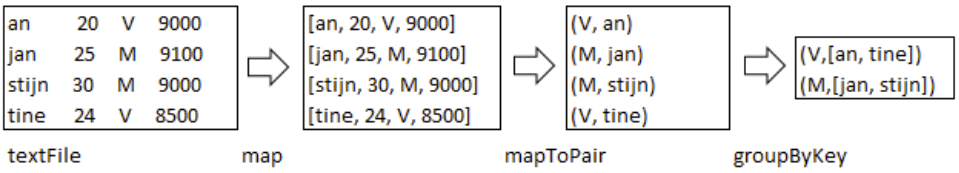
> The result is [(an,20_v), (jan,25_m), (stijn,30_m), (tine,24_v)]



groupByKey

```
package bdstudents.quickstart;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.*;
import scala.Tuple2;

public class App {
    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setAppName("Simple Application");
        JavaSparkContext sc = new JavaSparkContext(conf);

        JavaPairRDD<String, String> resultRDD =
        sc.textFile("/user/hduser/input/data")
        .map(line -> line.split("\t"))
        .mapToPair(fields -> new Tuple2<String, String>(fields[2],
fields[0]))
        .groupByKey();

        System.out.println(resultRDD.collect());
        sc.close();
    }
}
```

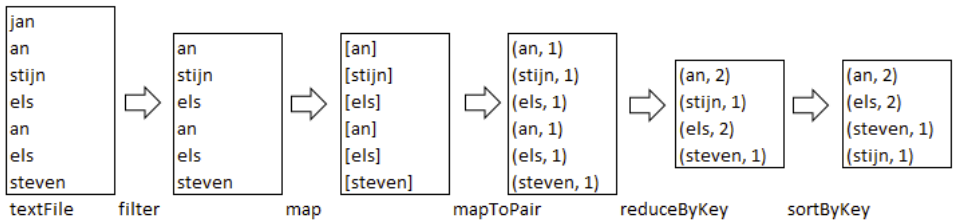
> The result is [(V,[an, tine]), (M,[jan, stijn])]



reduceByKey

```
package bdstudents.quickstart;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.*;
```

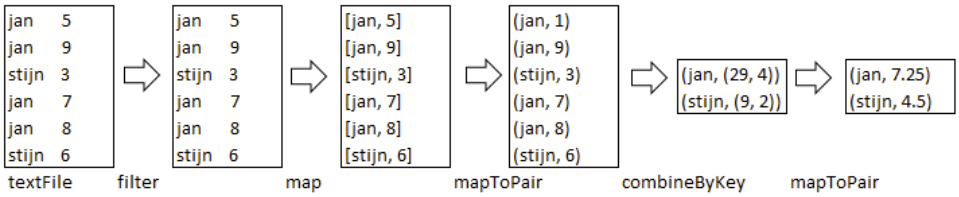
```
import scala.Tuple2;

public class App {
    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setAppName("Simple Application");
        JavaSparkContext sc = new JavaSparkContext(conf);

        JavaPairRDD<String, Integer> resultRDD =
sc.textFile("/user/hduser/input/names")
        .filter(line -> !line.startsWith("jan"))
        .map(line -> line.split("\t"))
        .mapToPair(fields -> new Tuple2<String, Integer>(fields[0], 1))
        .reduceByKey((x,y) -> x + y)
        .sortByKey();

        System.out.println(resultRDD.collect());
        sc.close();
    }
}
```



combineByKey

```
package bdstudents.quickstart;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.*;
import scala.Tuple2;

public class App {
    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setAppName("Simple Application");
        JavaSparkContext sc = new JavaSparkContext(conf);

        JavaPairRDD<String, Double> resultRDD =
sc.textFile("/user/hduser/input/scores")
        .filter(line -> line.length() > 0)
        .map(line -> line.split("\t"))
        .mapToPair(fields -> new Tuple2<String, Integer>(fields[0],
Integer.parseInt(fields[1])))
        .combineByKey(
            x -> new Tuple2<Integer, Integer>(x, 1),
            (x, y) -> new Tuple2<Integer, Integer>(x._1 + y, x._2 + 1),
            (x, y) -> new Tuple2<Integer, Integer>(x._1 + y._1, x._2 + y._2)
        )
        .mapToPair(t -> new Tuple2<String, Double>(t._1, (t._2._1 * 1.0) /
t._2._2));

        System.out.println(resultRDD.collect());
        sc.close();
    }
}
```

> The result is [(stijn,4.5), (jan,7.25)]



count

```
package bdstudents.quickstart;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.*;
public class App {
    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setAppName("Simple Application");
        JavaSparkContext sc = new JavaSparkContext(conf);

        long numberOfNonEmptyLines =
sc.textFile("/user/hduser/input/scores")
        .filter(line -> line.length() > 0)
        .count();

        System.out.println("Number of non empty lines: " +
numberOfNonEmptyLines);
        sc.close();
    }
}
```

> The result is Number of non empty lines: 6

countByValue

```
package bdstudents.quickstart;
import java.util.Map;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.*;
import scala.Tuple2;

public class App {
    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setAppName("Simple Application");
        JavaSparkContext sc = new JavaSparkContext(conf);

        Map<String, Long> result = sc.textFile("/user/hduser/input/scores")
        .filter(line -> line.length() > 0)
        .map(line -> line.split("\t"))
        .mapToPair(fields -> new Tuple2<String, String>(fields[0],
fields[1]))
        .keys()
        .countByValue();
```

```
        System.out.println(result);
        sc.close();
    }
}
```

> The result is {stijn=2, jan=4}

first

```
package bdstudents.quickstart;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.*;

public class App {
    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setAppName("Simple Application");
        JavaSparkContext sc = new JavaSparkContext(conf);

        String result = sc.textFile("/user/hduser/input/scores")
        .filter(line -> line.length() > 0)
        .first();

        System.out.println(result);
        sc.close();
    }
}
```

> The result is jan 5

max

```
package bdstudents.quickstart;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.*;

public class App {
    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setAppName("Simple Application");
        JavaSparkContext sc = new JavaSparkContext(conf);

        Integer result = sc.textFile("/user/hduser/input/scores")
        .filter(line -> line.length() > 0)
        .map(line -> Integer.parseInt((line.split("\t")[1])))
        .max((x,y) -> x.compareTo(y));

        System.out.println(result);
        sc.close();
    }
}
```

> The result is 9

take

```
package bdstudents.quickstart;
import java.util.List;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.*;

public class App {
    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setAppName("Simple Application");
        JavaSparkContext sc = new JavaSparkContext(conf);

        List<String> result = sc.textFile("/user/hduser/input/scores")
        .filter(line -> line.length() > 0).take(3);

        System.out.println(result);
        sc.close();
    }
}
```

> The result is [jan 5, jan 9, stijn 3]

top

```
package bdstudents.quickstart;
import java.util.List;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.*;

public class App {
    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setAppName("Simple Application");
        JavaSparkContext sc = new JavaSparkContext(conf);

        List<String> result = sc.textFile("/user/hduser/input/scores")
        .filter(line -> line.length() > 0).top(3);

        System.out.println(result);
        sc.close();
    }
}
```

> The result is [stijn 6, stijn 3, jan 9]

reduce

```
package bdstudents.quickstart;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.*;

public class App {
    public static void main(String[] args) {
```

```
        SparkConf conf = new SparkConf().setAppName("Simple Application");
        JavaSparkContext sc = new JavaSparkContext(conf);

        Integer result = sc.textFile("/user/hduser/input/scores")
        .filter(line -> line.length() > 0)
        .map(line -> Integer.parseInt((line.split("\t")[1])))
        .reduce((x,y) -> x + y);

        System.out.println(result);
        sc.close();
    }
}
```

> The result is 38

fold

```
package bdstudents.quickstart;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.*;

public class App {
    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setAppName("Simple Application");
        JavaSparkContext sc = new JavaSparkContext(conf);

        Integer result = sc.textFile("/user/hduser/input/scores")
        .filter(line -> line.length() > 0)
        .map(line -> Integer.parseInt((line.split("\t")[1])))
        .fold(0, (x,y) -> x + y);

        System.out.println(result);
        sc.close();
    }
}
```

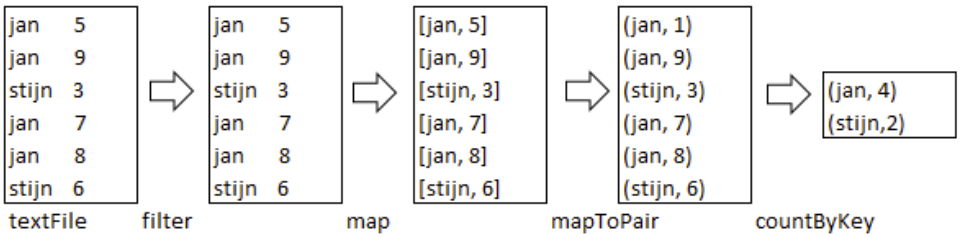
> The result is 38

countByKey

```
package bdstudents.quickstart;
import java.util.Map;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.*;
import scala.Tuple2;

public class App {
    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setAppName("Simple Application");
        JavaSparkContext sc = new JavaSparkContext(conf);

        Map<String, Long> result = sc.textFile("/user/hduser/input/scores")
        .filter(line -> line.length() > 0)
        .map(line -> line.split("\t"))
```

```
        .mapToPair(fields -> new Tuple2<String, Integer>(fields[0],
Integer.parseInt(fields[1])))
        .countByKey();

        System.out.println(result);
        sc.close();
    }
}
```

> The result is {stijn=2, jan=4}



lookup

```
package bdstudents.quickstart;
import java.util.List;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.*;
import scala.Tuple2;

public class App {
    public static void main(String[] args) {
        SparkConf conf = new SparkConf().setAppName("Simple Application");
        JavaSparkContext sc = new JavaSparkContext(conf);

        List <Integer> result = sc.textFile("/user/hduser/input/scores")
        .filter(line -> line.length() > 0)
        .map(line -> line.split("\t"))
        .mapToPair(fields -> new Tuple2<String, Integer>(fields[0],
Integer.parseInt(fields[1])))
        .lookup("stijn");

        System.out.println(result);
        sc.close();
    }
}
```

> The result is [3, 6]