Aufgabe 1

```
1  DROP TABLE IF EXISTS lists CASCADE;
2
3  CREATE TABLE lists (
4    id integer NOT NULL,
5    pos integer NOT NULL,
6    val float NOT NULL
7  );
8
9  INSERT INTO lists VALUES
10   (721, 1, 13.3),
11   (721, 4, 6.7),
12   (721, 5, 1.2),
13   (721, 2, 2.3),
14   (721, 3, 6.5),
15   (987, 5, 2.5),
16   (987, 3, 2.5),
17   (987, 1, 2.5),
18   (987, 2, 2.4),
19   (987, 4, 10.1),
20   (123, 1, 9);
21
22 -- retrieve the filenode
23 SELECT relname, relfilenode
24 FROM   pg_class
25 WHERE  relname='lists';
26 --
27 SHOW data_directory;
28 SELECT pg_relation_filepath('lists');
```

Aufgabe 2

- *Speicherhierarchie*
  Speicherhierarchie erreicht einen preiswerten Kompromiss zwischen Zugriffsgeschwindikeit und Speicherkapazität, indem eine durch Strategien und Heuristiken als aktuell signifikant befundene Teilmenge an Daten eines großen Speichermediums auf schnellem aber in der größe begrenzten Speicher bereit gestellt wird. Dies ist über mehrere Schichten möglich.

- *Seek time*
  Seek time bezeichnet die Verzögerungszeit, die bei einer Festplatte für das Bewegen des Schreib- und Lesearms zu einem anderen Track aufgebracht wird.

- *Rotational delay*
  Rotational delay bezeichnet die Verzögerungszeit, die bei einer Festplatte aufgebracht wird, um Daten unter den Schreib- Lesekopf zu rotieren.

- *Transfer time*
  Transfer time bezeichnet die Übertragungsdauer des grundlegenden Lese- oder Schreibprozesses einer Festplatte.

Aufgabe 3 Verwaltet ein DBMS seinen Speicher selbst ist es nicht weiter durch

Gegebenheiten des Betriebssystems eingeschränkt. Der Speicherbereich lässt sich dann frei, auch mit Hilfe von anderen Managern optimal für die jeweiligen Anforderungen verwalten, was Performance Vorteile birgt. .

Aufgabe 4

1. $t = t_s + t_r + t_{tr} = 3.4\,\text{ms} + \frac{1}{2} \cdot \frac{60}{15\,000\,\frac{1}{\text{s}}} + \frac{8\,\text{kB}}{163\,\frac{\text{MB}}{\text{s}}} = 5.45\,\text{ms}$

   - *random access*
     $t_{rdm} = 10.000 \cdot 5.45\,\text{ms} = 54.5\,\text{ms}$

   - *sequential read*
     Quelle Folien Storage, Seite 7:
     Track to track: $t_{tt} = 0.2\,\text{ms}$; Daten a track: 512 kB, damit $\lceil \frac{10000 \cdot 8\,\text{kB}}{512\,\text{kB}} \rceil = 157$ Tracks
     $t_{seq} = t_s + t_r + 10000 \cdot t_{tr} + 157 \cdot t_{tt} = 536.9\,\text{ms}$

2. Sequentieller Zugriff erfordert nur einmaligen Aufwand von t_s und t_r. Bis auf die 157 Track to Track seeks wird konsequent gelesen.