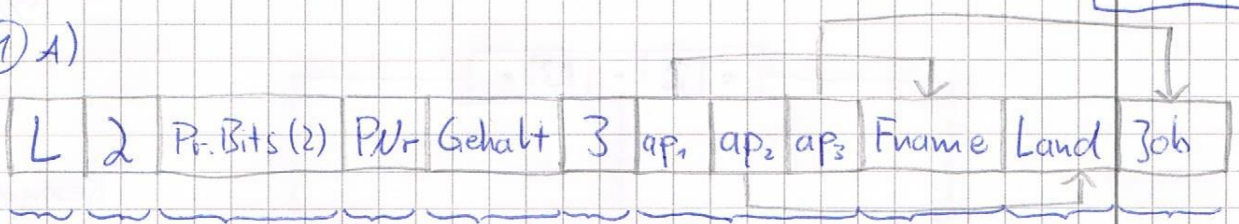
① A)



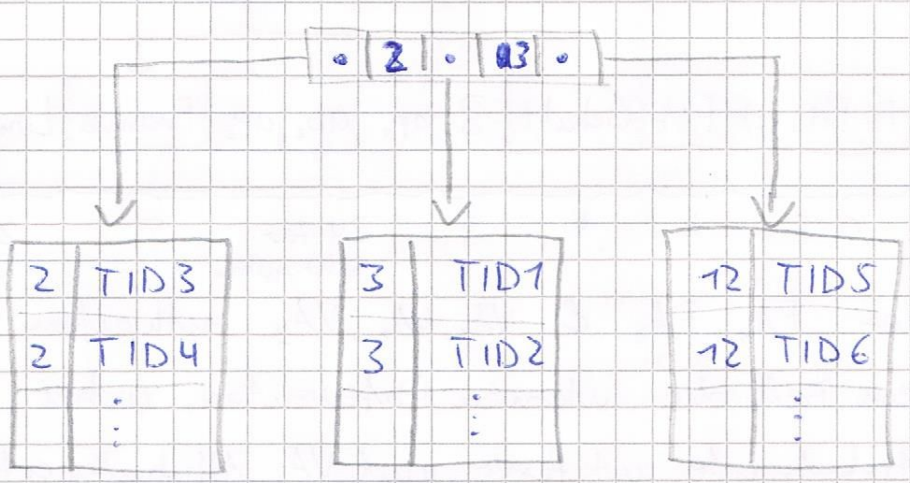
Bytes: 4 | 4 | 0,25 | 4 | 4 | 4 | 24 für 64 Bit System | 5 | 2 | 9

- Die Felder FA_1, FA_2 & VA_1, VA_2, VA_3 enthalten in diesem Fall die internen Datensätze, wobei FA_i Attributwerte mit fixer und VA_i Attributwerte mit variabler Länge speichert.
- Der Datensatz benötigt 60,25 -> 61 Byte (64 Bit System) bzw. 48,25 -> 49 Byte (32 Bit System)
- Die Nutzdaten nehmen dabei 24 Byte in Anspruch.
- Um den Platzbedarf zu reduzieren könnte man den Datensatz in die 3NF bringen, wodurch die vielen Jobs: varchar(x) Einträge nur noch einmal gespeichert werden müssen.

B) - Die TID's lauten 9876.1, 9876.2, 9876.3, 9876.4 und 9876.5.
- Die Offsetpositionen wären dann 900, 800, 700, 600 und 500. Dabei wurde berücksichtigt, das ein Datensatz nur 37 Byte groß sind passt jeder in ein 100 Byte großen Slot. Der überschüssige Platz von 63 Byte bietet Raum für Veränderungen der absatz Tabelle.
- Satztabelle

1	9876.1
2	9876.2
3	9876.3
4	9876.4
5	9876.5

C) - CREATE INDEX absatz_pnr_ix ON absatz ([redacted]);
PNr Sales



Besonderheit:
Der Baum hat Indexe als Blätter

D) - ALTER TABLE absatz
ADD Kommentar char(256);

- Da der neue Eintrag nicht in die freien Speicherräume passt, werden die Datensätze verschoben. Die TID's bleiben dabei unverändert, allerdings zeigen diese jetzt nicht direkt auf den Datensatz, sondern dort wo der Datensatz zwar war steht jetzt eine Adresse im Speicher, welche zu der neuen Position des Datensatzes verweist. Der Suchbaum aus C) ändert sich dabei nicht (da sich die Satztabelle nicht ändert).