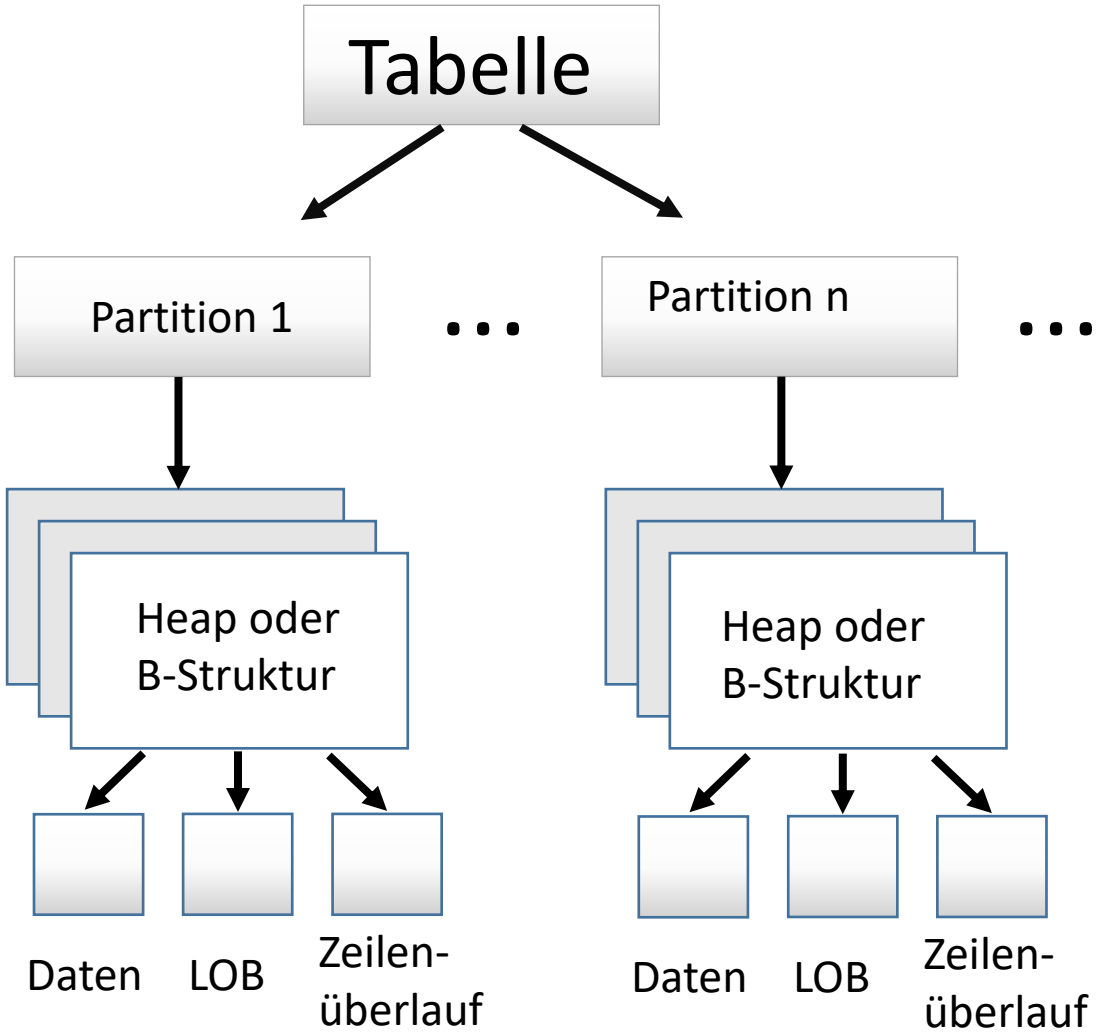
Indizes

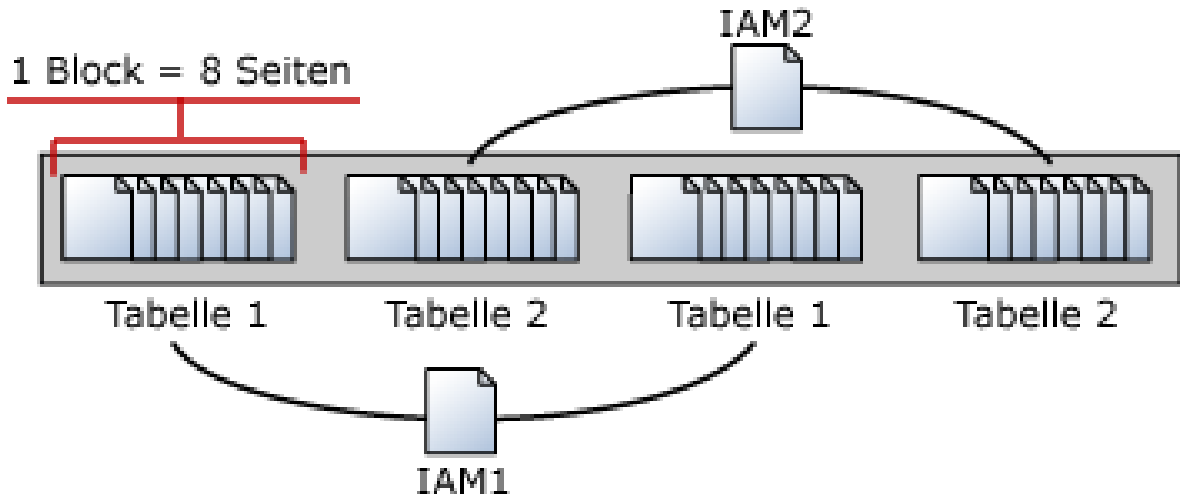
Speicherstruktur



- Tabellen- und Indexseiten in einer / mehrerer Partitionen enthalten
- Daten in Form von Heaps, sortiert oder als B-Struktur abgelegt
- Physisch als Datendatei und Transaktionsprotokolldatei gespeichert (*.mdf, *.ldf)

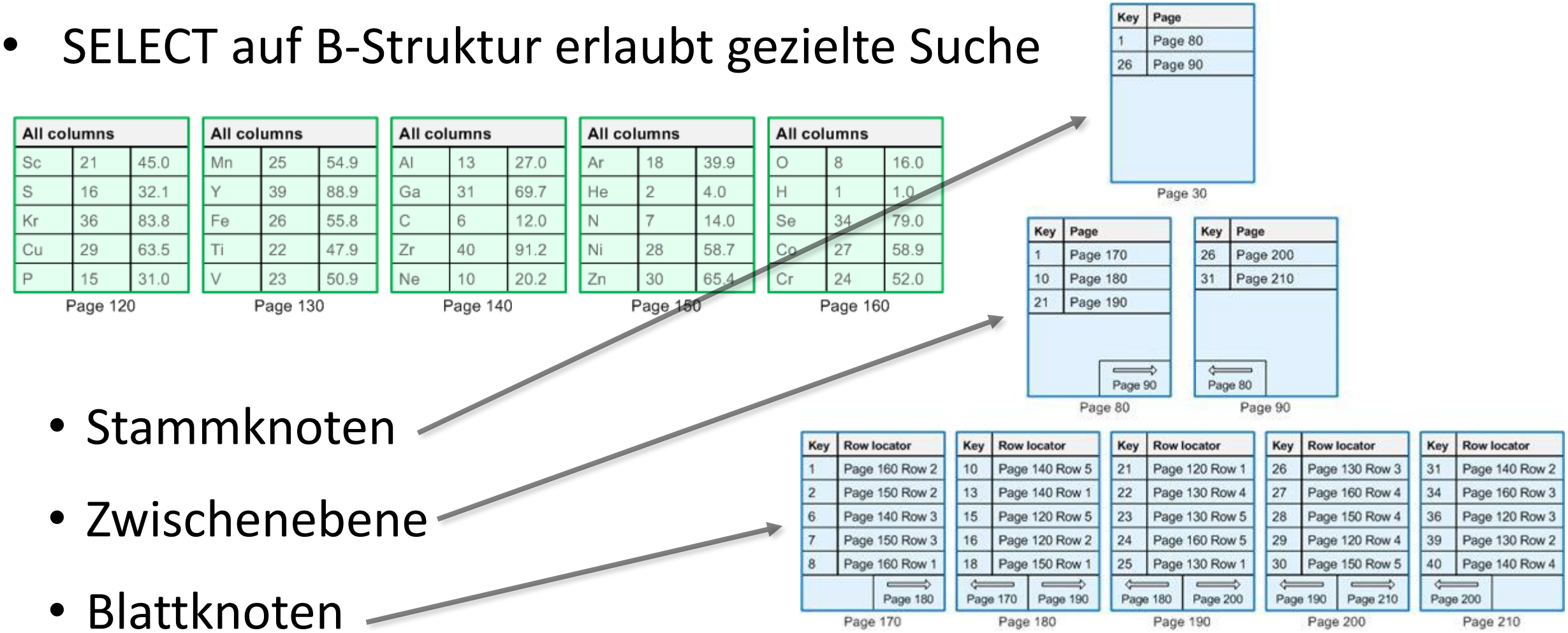
Heap

- Ansammlung von unsortierten Daten („Haufen“)
- Tabellen ohne Index nennt man Heaps
- Heaps führen zu Table-Scans
- SELECT auf einem Heap zwingt den SQL Server zu einer linearen Suche
- Seiten werden ohne Reihenfolge gespeichert



B-Struktur

- SELECT auf B-Struktur erlaubt gezielte Suche



- Stammknoten
- Zwischenebene
- Blattknoten

Index

- Bei Zuweisung eines PRIMARY KEY‘s wird automatisch ein Index für diese Tabelle erstellt
- Nur ein gruppierter Index pro Tabelle möglich, da auch nur nach einer Reihenfolge sortiert werden kann
- In MS SQL Server gibt es folgende Indizes:
  - Gruppiert
  - Nicht gruppiert
  - Eindeutig

Gruppierter Index

- Sortierte physische Speicherung der Datenzeilen
- Sortierung anhand des Index-Keys
  - ➢ Primary Key-Spalte nicht immer sinnvoll; Indizierung sinnvoll wählen!
  - ➢ Nach was wird am häufigsten gesucht? (Bestell-nummer[id] oder Bestelldatum?)

Nicht gruppierter Index

- unsortierte physische Speicherung der Datenzeilen
- Heap oder gruppierter Index liegt zugrunde
  - ➢ Bis zu 999 nicht gruppierte Indizes pro Tabelle möglich
- Wird durch Verweisen auf das Original anhand von Spalteninformationen erstellt (Zeilenlokator)

Syntax

```
CREATE [UNIQUE][CLUSTERED | NONCLUSTERED] INDEX index_name
 ON <object> ( column [ ASC | DESC ] [ ,...n ] )
[INCLUDE ( column_name [ ,...n ] ) ]
[WHERE <filter_predicate> ]
[WITH ( <relational_index_option> [ ,...n ] ) ]
[ ; ]
```

UNIQUE Einschränkung bewirkt Überprüfen auf Eindeutigkeit des Indexschlüssels; doppelte Einträge werden nicht zugelassen

WITH-Options

WITH-Option	Zweck
ALLOW_ROW_LOCKS	Aktiviert bzw. deaktiviert Sperrungen auf Zeilenebene für Indizes
ALLOW_PAGE_LOCKS	Aktiviert bzw. deaktiviert Sperrungen auf Seitenebene für Indizes
ONLINE	Aktiviert bzw. deaktiviert den Zugriff auf Indizes während der Erstellung
FILLFACTOR	Verwaltet freien Speicherplatz von Seiten auf Blattebene
PAD_INDEX	Verwaltet freien Speicherplatz von Seiten auf Nicht-Blattebene

Gefilterte Indizes

- Um Platz zu sparen kann es in einigen Fällen von Vorteil sein, nicht alle Werte einer Spalte zu indizieren
- In solchen Fällen kann man zu einem gefilterten Index greifen

```
CREATE NONCLUSTERED INDEX IX_BlablaIndex
ON Products (OrderDate)
WHERE (OrderDate > ’01.01.1999‘);
```

Freien Speicher im Index festlegen

- Verfügbarer Speicherplatz in den Index-Seiten wirkt sich auf Leistung aus
- FILLFACTOR legt den verfügbaren Speicherplatz fest (in %)
  - ➢ Niedriger FILLFACTOR für hohe Transaktionslast
  - ➢ Hoher FILLFACTOR für hohe Abfragelast
- PAD_INDEX wendet FILLFACTOR auch auf Zwischenebenenseiten an

```
CREATE UNIQUE NONCLUSTERED INDEX IX_BlablaIndex
ON Products (OrderDate ASC)
WITH (FILLFACTOR = 65, PAD_INDEX = ON);
```

Weitere Informationen

- Löschen eines Index mit DROP

```
DROP INDEX IX_BlablaIndex ON <tabelle>;
```

- Ansehen, auf welchen Seiten der DB was gepeichert ist
  - `SELECT sys.fn_PhysLocFormatter(%%physloc%%) AS Location, * FROM Orders;`

- Links zum Thema:
  - http://blog.fumus.de/sql-server/2011/04/sql-server-index-leitfaden-indizes-fr-rookies-teil-1/
  - http://blog.fumus.de/sql-server/2011/04/sql-server-index-leitfaden-teil-2-vergabe-von-indizes/