Modul 153

Performance der Datenbank optimieren

Zugriffe Optimieren

Der Befehl EXPLAIN

- Mit **EXPLAIN** kann man die Strategie des Systems beim Ausführen einer Abfrage anzeigen lassen.

  `explain select ......`


```
CREATE TABLE kunde (
  id int(10) NOT NULL,
  nachname varchar(50) DEFAULT NULL,
  vorname varchar(50) DEFAULT NULL,
  strasse varchar(100) DEFAULT NULL,
  ort_id int(10) DEFAULT NULL,
  PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

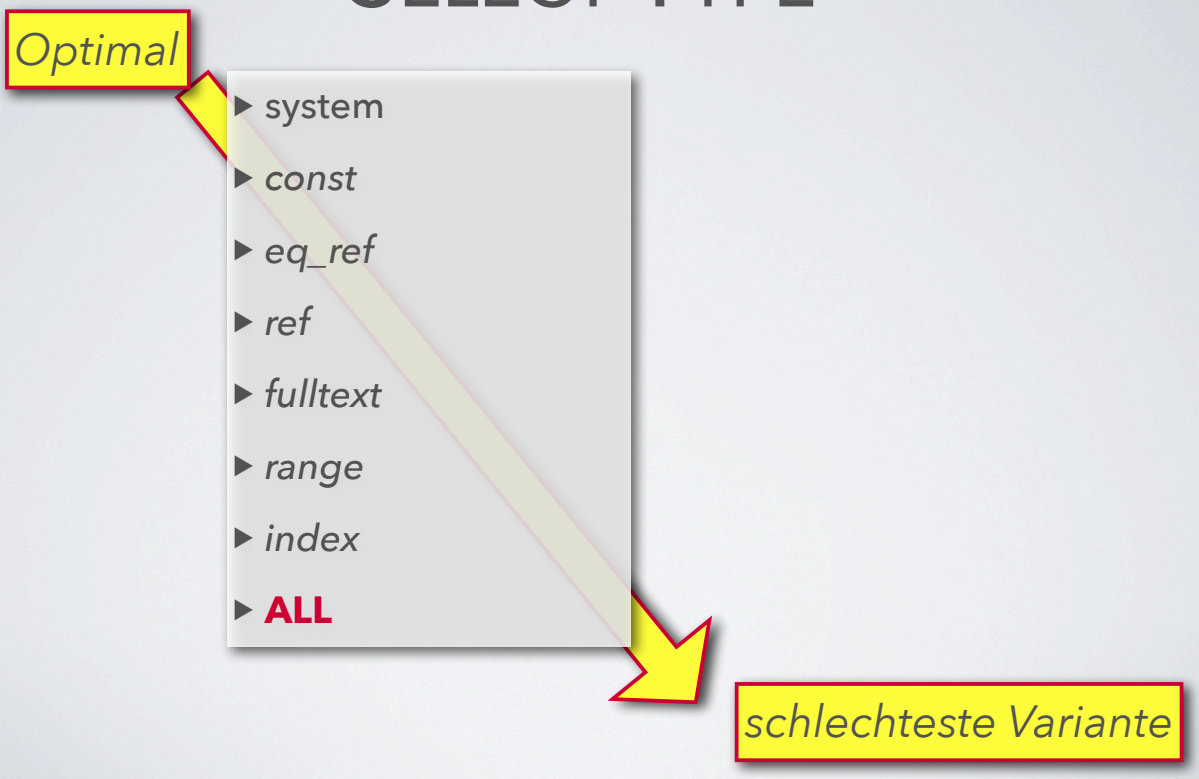
```
CREATE TABLE ort (
  id int(10) NOT NULL,
  plz varchar(255) DEFAULT NULL,
  plz_zusatz varchar(255) DEFAULT NULL,
  ort varchar(255) DEFAULT NULL,
  PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
mysql> explain select kunde.id, kunde.nachname, kunde.vorname, ort.ort from kunde, ort
              where kunde.ort_id = ort.id and nachname like 'Mei%';
+----+-------------+-------+--------+---------------+---------+---------+---------------------------------+------+-------------+
| id | select_type | table | type   | possible_keys | key     | key_len | ref                             | rows | Extra       |
+----+-------------+-------+--------+---------------+---------+---------+---------------------------------+------+-------------+
|  1 | SIMPLE      | kunde | ALL    | NULL          | NULL    | NULL    | NULL                            |  155 | Using where |
|  1 | SIMPLE      | ort   | eq_ref | PRIMARY       | PRIMARY | 4       | m153_performance.kunde.ort_id |    1 | NULL        |
+----+-------------+-------+--------+---------------+---------+---------+---------------------------------+------+-------------+
2 rows in set (0,00 sec)
```

```
alter table kunde add index nachname_idx (nachname);
```

```
mysql> explain select kunde.id, kunde.nachname, kunde.vorname, ort.ort from kunde, ort where kunde.ort_id = ort.id and nachname like 'Mei%';
+----+-------------+-------+--------+---------------+--------------+---------+---------------------------------+------+--------------------------------------+
| id | select_type | table | type   | possible_keys | key          | key_len | ref                             | rows | Extra                                |
+----+-------------+-------+--------+---------------+--------------+---------+---------------------------------+------+--------------------------------------+
|  1 | SIMPLE      | kunde | range  | nachname_idx  | nachname_idx | 153     | NULL                            |    3 | Using index condition; Using where |
|  1 | SIMPLE      | ort   | eq_ref | PRIMARY       | PRIMARY      | 4       | m153_performance.kunde.ort_id |    1 | NULL                                 |
+----+-------------+-------+--------+---------------+--------------+---------+---------------------------------+------+--------------------------------------+
2 rows in set (0,00 sec)
```

Select-Type



Denormalisieren

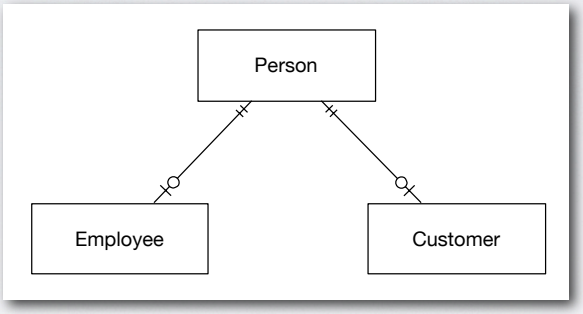
- Datenbanken sollten eigentlich sauber in der 3. Normalform vorliegen.

aber...

- Manchmal ist Denormalisieren notwendig, um gute Performance zu erhalten

Noch einmal: Vererbung

Korrekt normalisiert:
Pro Klasse eine Tabelle



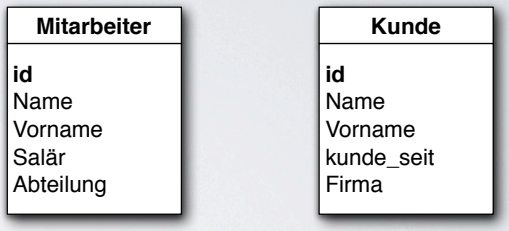
Nachteil: immer (mindestens) zwei
DB-Zugriffe notwendig

Lösung: De-Normalisierung

- Es gibt zwei gute Möglichkeiten, hier eine Verbesserung zu erreichen:
  - ➟ Horizontales Mapping
  - ➟ Single-Table Mapping

De-Normalisierung Variante 1

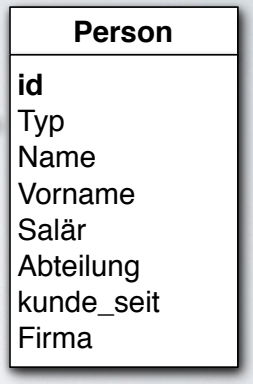
Horizontales Mapping



- **Vorteile**
  - ➟ Einfaches Hinzufügen weiterer Unterklassen
  - ➟ Ein DB-Zugriff genügt
- **Nachteile**
  - ➟ Strukturelle Redundanzen
  - ➟ Schwierig, neues Attribut für Oberklasse zu definieren

De-Normalisierung Variante 2

Single Table Mapping



- **Vorteile**
  - ➟ Zugriff schnell auf Ober- und Unterklasse
- **Nachteile**
  - ➟ Stark denormalisiert
  - ➟ Zusätzliches Attribut notwendig

Übungen

- Übung 29, Indices benutzen
- Schauen Sie sich einmal die komplexeren Abfragen aus den Modulen 104 und 105 mit dem Befehl explain an.