LU07 - Vertiefte SQL-Abfragen

Lernziele

- Ich kenne alle Vergleichsoperatoren und kann sie in WHERE einsetzen.
- Ich kann mit BETWEEN und IN Bedingungen kürzer formulieren, die ich sonst mit AND/OR schreiben müsste.
- Ich kann mit LIKE nach Textmustern suchen.
- Ich kann mit IS NULL prüfen, ob ein Wert fehlt.
- Ich kann mit DISTINCT doppelte Werte entfernen und mit LIMIT die Ergebnismenge einschränken.

Einleitung

In LU02 haben Sie gelernt, mit SELECT, FROM, WHERE und ORDER BY Daten abzufragen. Heute erweitern wir vor allem das WHERE: mit mehr Operatoren, mit Textmustern und mit dem Sonderfall „kein Wert vorhanden“. Ausserdem lernen Sie zwei neue Werkzeuge, um die Ergebnismenge zu formen: DISTINCT und LIMIT.

Vergleichsoperatoren

In LU02 haben Sie bereits mit > gearbeitet. Hier die vollständige Liste:

```
= , < , > , <= , >= , <>
```

```
SELECT Series_Title, Released_Year
FROM imdb_top_1000
WHERE Series_Title = 'Interstellar';
```

Sie kennen die Regel bereits: Text steht in Anführungszeichen, Zahlen nicht. <> bedeutet **ungleich**.

```
-- Filme vor 1970
SELECT Series_Title, Released_Year
FROM imdb_top_1000
WHERE Released_Year < 1970;
```

BETWEEN

Prüft, ob ein Wert zwischen zwei Grenzen liegt – **beide Grenzen eingeschlossen**.

```
-- Filme aus den 90ern
SELECT Series_Title, Released_Year
FROM imdb_top_1000
WHERE Released_Year BETWEEN 1990 AND 1999;
```

Das ist dasselbe wie:

```
WHERE Released_Year >= 1990 AND Released_Year <= 1999
```

BETWEEN ist also keine neue Fähigkeit, sondern eine **kürzere Schreibweise** für etwas, das Sie mit AND bereits könnten.

IN

Prüft, ob ein Wert in einer **Liste** vorkommt.

```
-- Filme von Nolan oder Coppola
SELECT Series_Title, Director
FROM imdb_top_1000
WHERE Director IN ('Christopher Nolan', 'Sofia Coppola');
```

Das ist dasselbe wie:

```
WHERE Director = 'Christopher Nolan' OR Director = 'Sofia
Coppola'
```

Auch hier gilt: IN ersetzt eine Kette von OR-Bedingungen. Bei zwei Werten ist der Unterschied klein, bei fünf oder sechs Regisseur:innen macht IN die Abfrage deutlich lesbarer.

LIKE

Mit LIKE suchen Sie nach **Textmustern** statt nach exakten Werten.

- % = beliebig viele Zeichen (auch null)
- _ = genau ein Zeichen

```
-- Alle Filme, die mit "The" beginnen
SELECT Series_Title
```

```
FROM imdb_top_1000
WHERE Series_Title LIKE 'The%';
```

```
-- Alle Filme, die "Star" im Titel enthalten
SELECT Series_Title
FROM imdb_top_1000
WHERE Series_Title LIKE '%Star%';
```

Der Unterstrich steht für genau ein beliebiges Zeichen – nützlich, wenn Sie die Länge kennen, aber nicht jeden Buchstaben:

```
-- Ein vierbuchstabiger Titel, der mit "C" beginnt
SELECT Series_Title
FROM imdb_top_1000
WHERE Series_Title LIKE 'C___';
```

IS NULL / IS NOT NULL

NULL bedeutet in SQL: **kein Wert vorhanden** – nicht dasselbe wie 0 oder ein leerer Text.

Manche Filme haben keinen Meta_score (die Bewertung von professionellen Kritiker:innen auf Metacritic, die Sie schon aus der LU02-Übung kennen):

```
-- Filme ohne Meta_score
SELECT Series_Title, Meta_score
FROM imdb_top_1000
WHERE Meta_score IS NULL;
```

```
-- Filme mit Meta_score
SELECT Series_Title, Meta_score
FROM imdb_top_1000
WHERE Meta_score IS NOT NULL;
```

`WHERE Meta_score = NULL` funktioniert **nicht** – Gleichheit ist für NULL nicht definiert. Es muss immer IS NULL bzw. IS NOT NULL heissen.

Bedingungen kombinieren

AND, OR und NOT kennen Sie bereits aus LU02. Jetzt kombinieren Sie sie mit den neuen Operatoren:

```
-- Alle Sci-Fi-Filme von Christopher Nolan
SELECT Series_Title, Genre, Director
```

```
FROM imdb_top_1000
WHERE Director = 'Christopher Nolan'
  AND Genre LIKE '%Sci-Fi%';
```

```
-- Alle Filme, die KEINE Komödie sind
SELECT Series_Title, Genre
FROM imdb_top_1000
WHERE NOT Genre LIKE '%Comedy%';
```

Auch BETWEEN und IN lassen sich mit AND/OR kombinieren:

```
-- Filme der 90er mit einem Rating über 8.5
SELECT Series_Title, Released_Year, IMDB_Rating
FROM imdb_top_1000
WHERE Released_Year BETWEEN 1990 AND 1999
  AND IMDB_Rating > 8.5;
```

DISTINCT

Entfernt doppelte Werte aus dem Ergebnis. Beispiel: Die Spalte Certificate (Altersfreigaben wie „U“, „PG-13“, „R“) kommt in der Tabelle viele Male vor.

```
-- Alle vorkommenden Altersfreigaben, je einmal
SELECT DISTINCT Certificate
FROM imdb_top_1000;
```

Ohne DISTINCT würde jede Certificate-Ausprägung so oft erscheinen, wie sie in der Tabelle vorkommt.

LIMIT

Schränkt die Anzahl der zurückgegebenen Zeilen ein.

```
-- Die 5 bestbewerteten Filme
SELECT Series_Title, IMDB_Rating
FROM imdb_top_1000
ORDER BY IMDB_Rating DESC
LIMIT 5;
```

LIMIT ohne ORDER BY liefert eine zufällige Auswahl. MySQL garantiert ohne ORDER BY keine bestimmte Reihenfolge der Zeilen – „die ersten 5“ gibt es ohne Sortierung nicht wirklich, auch wenn immer wieder dieselben 5 erscheinen

können. Verwenden Sie LIMIT deshalb fast immer zusammen mit ORDER BY.

Zusammenfassung

Operator	**Beispiel**
= , < , > , ⇐ , >= , <>	*IMDB_Rating > 9.0*
BETWEEN ... AND ...	*Released_Year BETWEEN 1990 AND 1999*
IN (...)	*Director IN ('Nolan','Coppola')*
LIKE	*Series_Title LIKE 'The%'*
IS NULL / IS NOT NULL	*Meta_score IS NULL*
AND / OR / NOT	*Director = 'Nolan' AND Genre LIKE '%Sci-Fi%'*
DISTINCT	*SELECT DISTINCT Certificate* – keine doppelten Ergebnisse
LIMIT	*LIMIT 5* – nur die ersten 5 Ergebnisse (mit ORDER BY!)

M290-LU07

Guido Koch