SQL
Verknüpfung von Tabellen

Welche Bestellungen hat Kunde ... aufgegeben?

Welche Kunden im Staat USA haben Waren im Wert von mindestens ... bestellt?

Welche Waren wurden nicht bestellt?

Relationales Datenbankmodell

- Ablage von Daten in Tabellen. Für jede abzubildende Objektgruppe wird eine Tabelle angelegt.
- Erstellungen von Beziehungen (Relationen) zwischen Tabellen. Objektgruppen interagieren miteinander. Anzeige von Details zu einem Vorgang.
- Abbildung eines Enity Relationship-Modell.

Enity Relationship (ER) - Model

- Gegenstands-Beziehungsmodell
- Abbildungen von Gegenständen (Entity, Entität) aus der realen Welt.
- Darstellung von Objekten und deren Attributen.
- Visualisierung von Objekten und deren Beziehungen bezüglich eines Prozesses.

Entität (Gegenstand)

Album

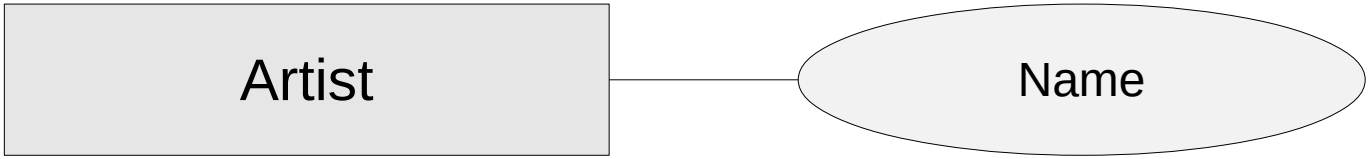
Artist

- Gegenstände werden mit Hilfe eines Rechtecks dargestellt.
- Darstellung von Substantiven in Textbeschreibungen.
- Jede Entität symbolisiert eine Tabelle in einer Datenbank.

Weitere Beispiele für Entitäten

- Gegenstände oder Lebewesen wie zum Beispiel Buchungsrechner, Server, Prospekt, Mensch.
- Existierende Strukturen wie zum Beispiel Personaldaten.
- Rollen von Menschen und Systemen wie zum Beispiel Teilnehmende, Projektleitende, Ausleihende.
- Organisatorische Einheiten wie zum Beispiel GmbH, Verkaufsgebiet, Ort.

Attribute einer Entity

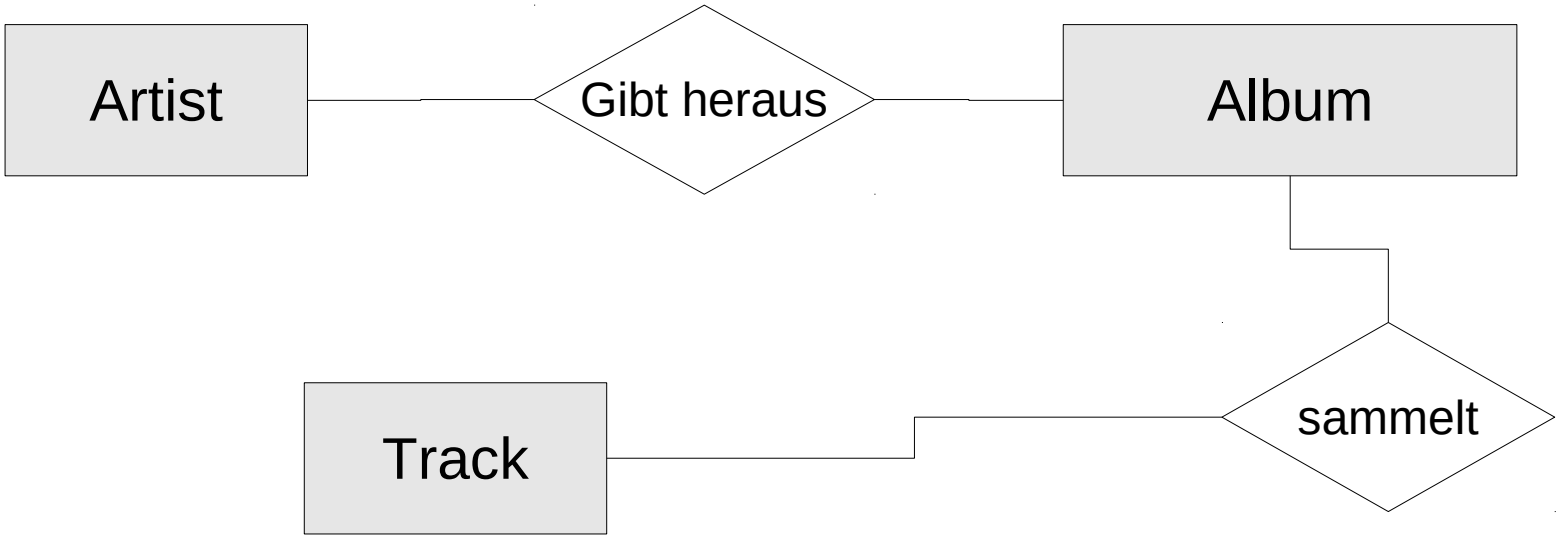


- Allgemeine Beschreibung einer Entität.
- Eigenschaft eines Objekts.
- Darstellung als Ellipse.
- Für jedes Attribut wird eine Spalte in der dazugehörigen Tabelle angelegt.

ID einer Entität

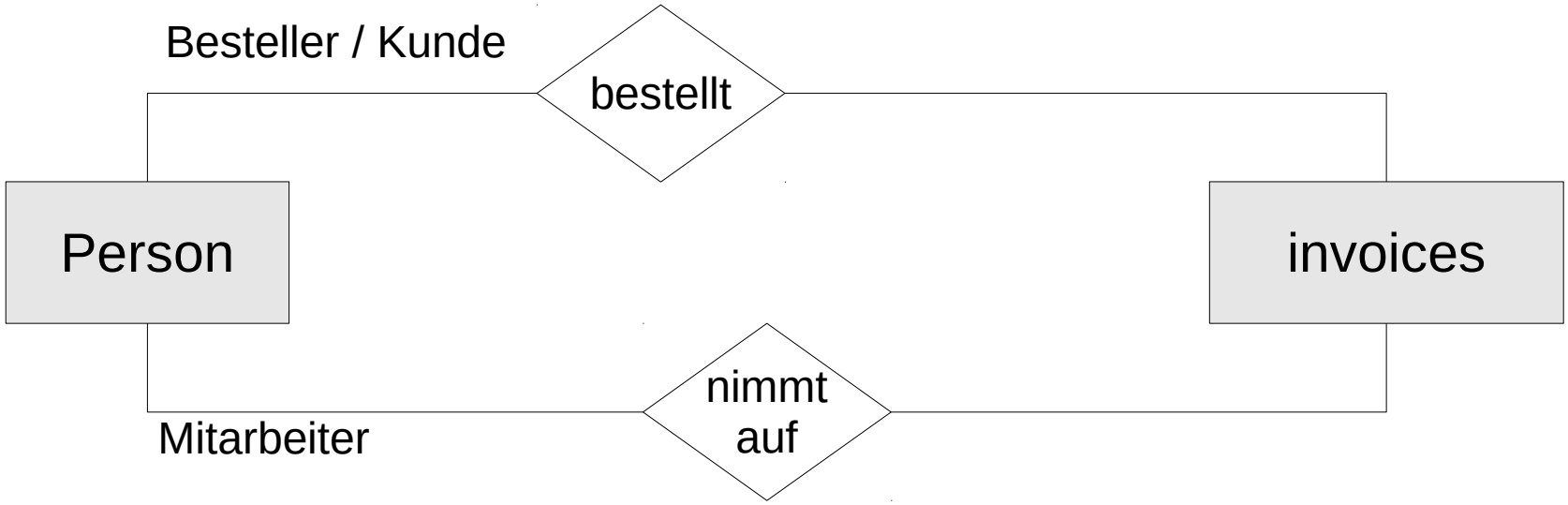
- Ids sind der „Hausschlüssel“ zu einem Gegenstand.
- Schlüssel identifizieren ein Objekt eindeutig.
- Schlüssel werden häufig durch ein künstliches Attribut abgebildet. Künstliche Attribut sind zum Beispiel die Personalnummer, die Ausweisnummer, die Artikelnummer.
- Namen von Ids werden im Entity-Relationship-Modell unterstrichen.

Relationship



- Die Raute verbindet zwei Entitäten.
- In der Raute wird die Beziehung mit Hilfe eines Verbs beschrieben.

Rollen in einer Beziehung

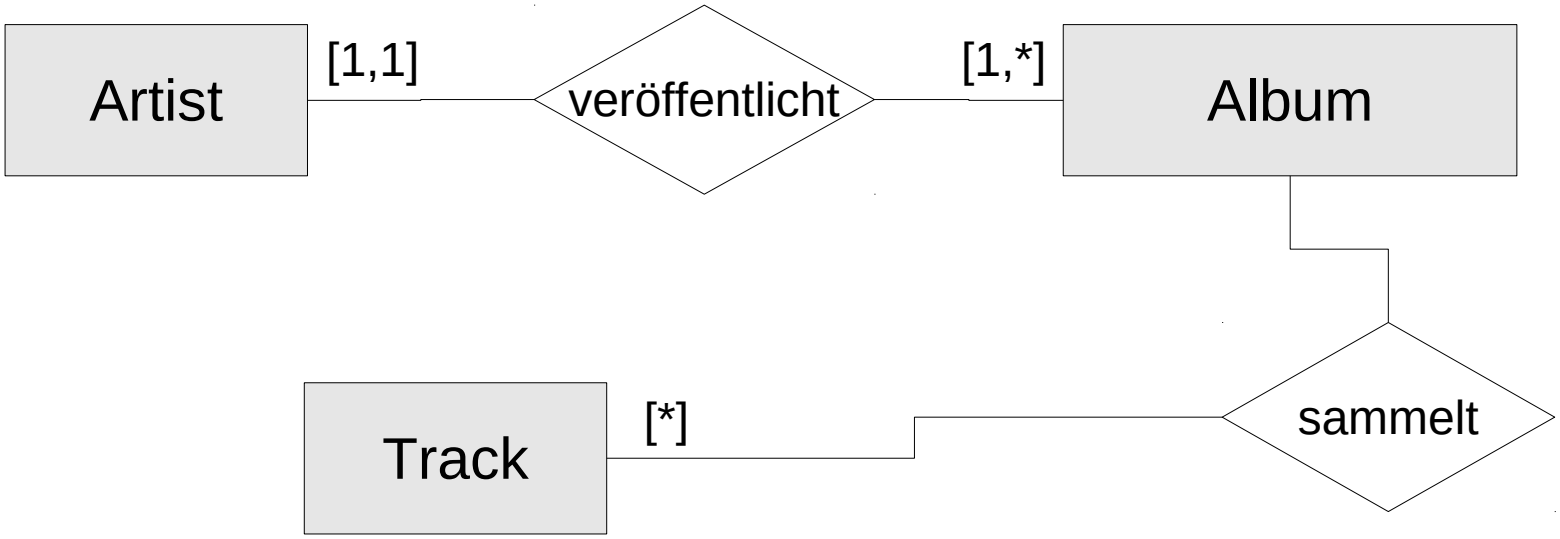


- Entitäten können verschiedene Rollen in einer Beziehung haben.
- Die jeweiligen Rollen werden an den Enden der Beziehungslinie gesetzt.

Rollen

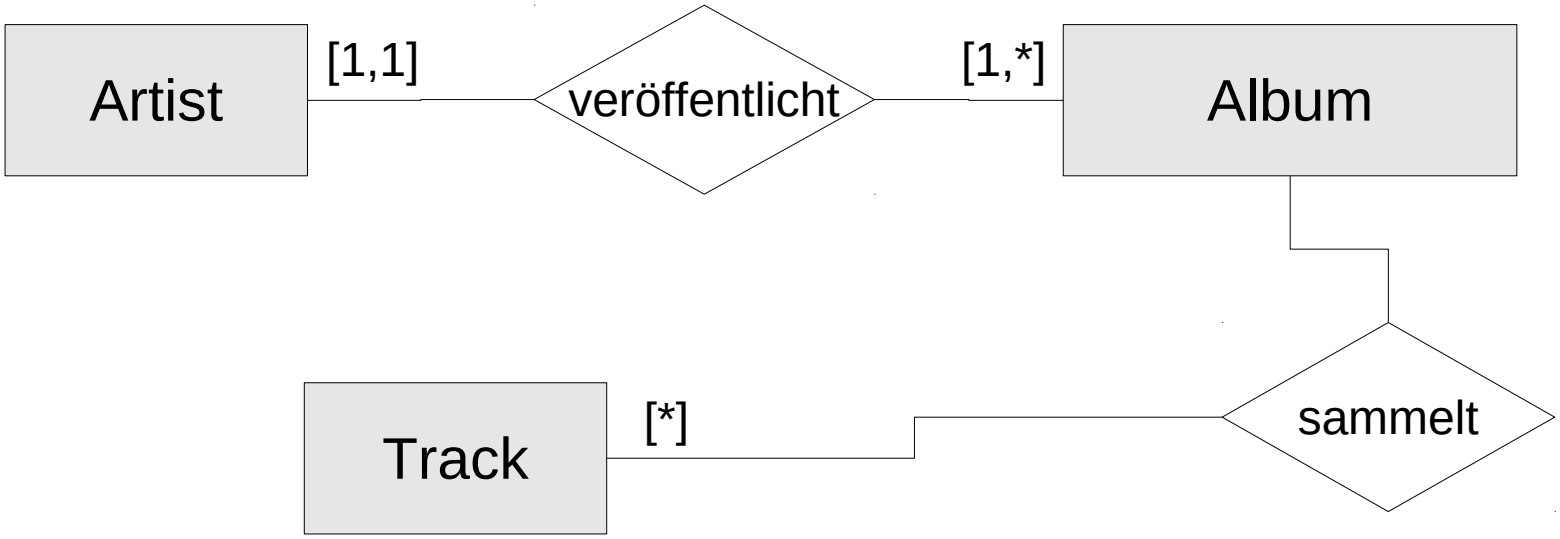
- Funktion, die ein Objekt in der abgebildeten Beziehung, erfüllt.
- Funktionen und Positionen in einer Hierarchie.

Kardinalitäten



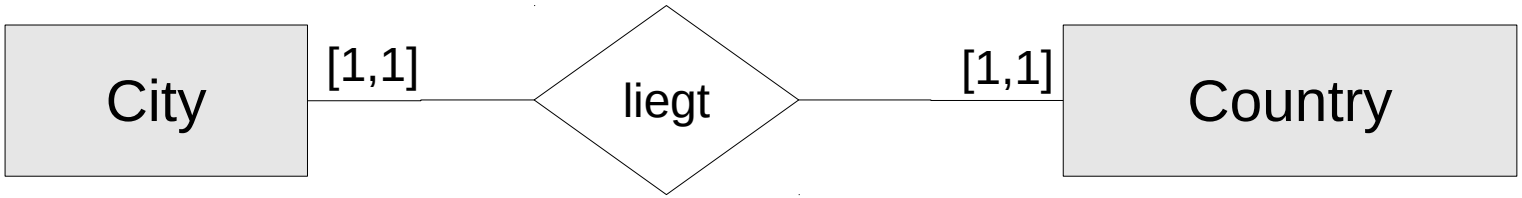
- Anzahl der beteiligten Entitäten.
- [min, max]-Notation der Kardinalität.
- Als Abkürzung wird häufig nur [max] angegeben.

Beispiel



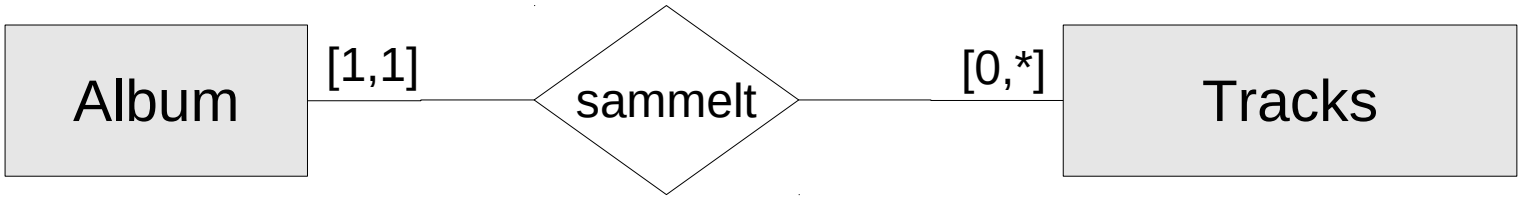
- In diesem Beispiel sammelt ein Album beliebig viele Tracks.
- Ein Kunstschaffender hat mindestens ein Album herausgegeben. Die Anzahl der herausgegebenen Alben ist aber nicht limitiert.

1:1-Relation



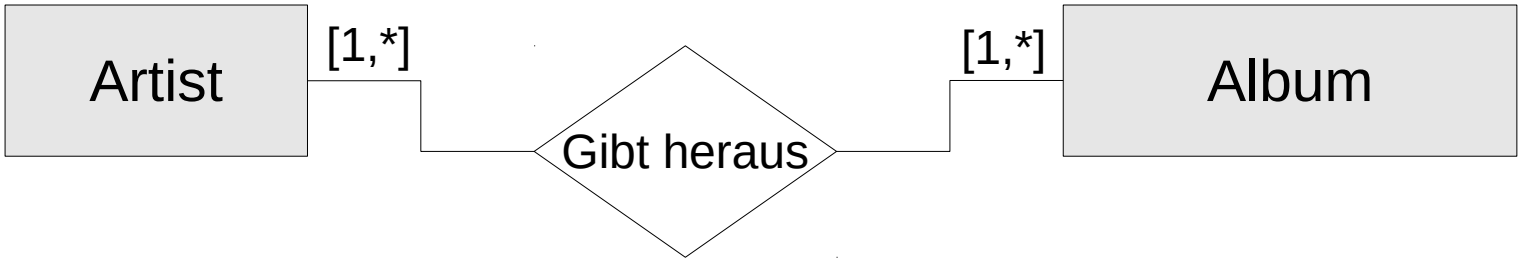
- Aussage: Eine Stadt liegt in einem Land.
- Einer Entität kann exakt einer anderen Entität zugeordnet werden.
- Die Informationen werden häufig in einer Tabelle angezeigt.

1:n-Relation



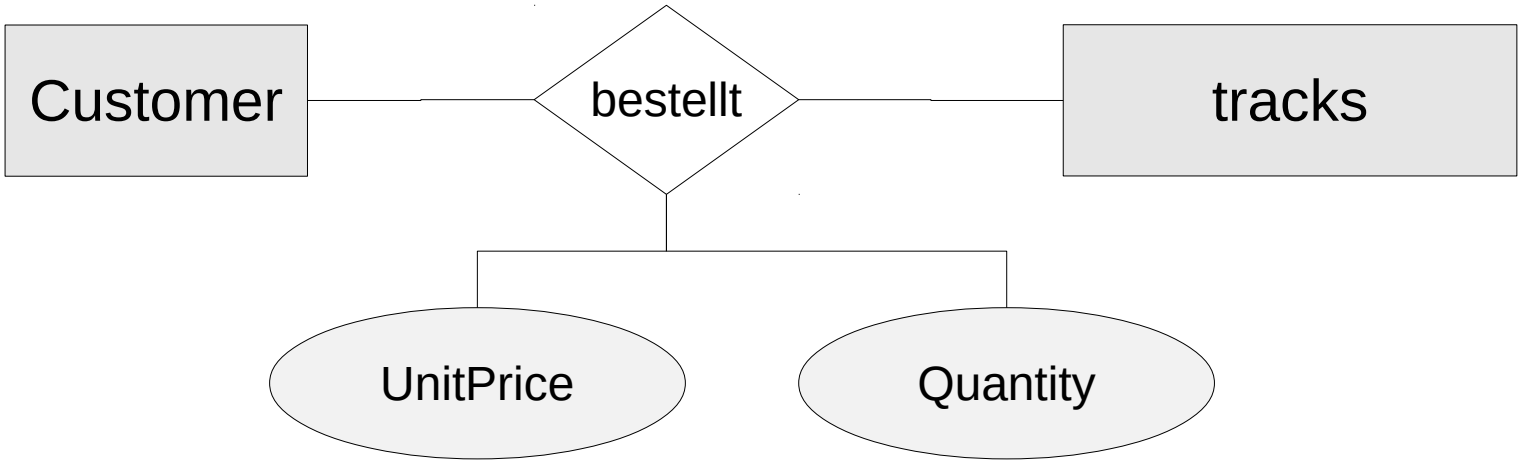
- Aussage: Ein Album kann aus beliebig vielen Musikstücken bestehen.
- Einer Entität können beliebig viele andere Entitäten zugeordnet werden.
- Die Relation wird am häufigsten mit Hilfe von zwei Tabellen abgebildet. In einer der beiden Tabellen wird mit Hilfe eines Attributs auf die übergeordnete Entität verwiesen.

m:n-Relation



- Aussage: Beliebig viele Kunstschaffende können beliebig viele Alben herausgeben.
- Die Relation wird in einer Pseudo-Tabelle abgebildet, die die Beziehung beschreibt. Für jede Entität wird eine Spalte angelegt. Mit Hilfe des Schlüssels wird auf die gewünschte Entität verwiesen.

Attribute einer Beziehung

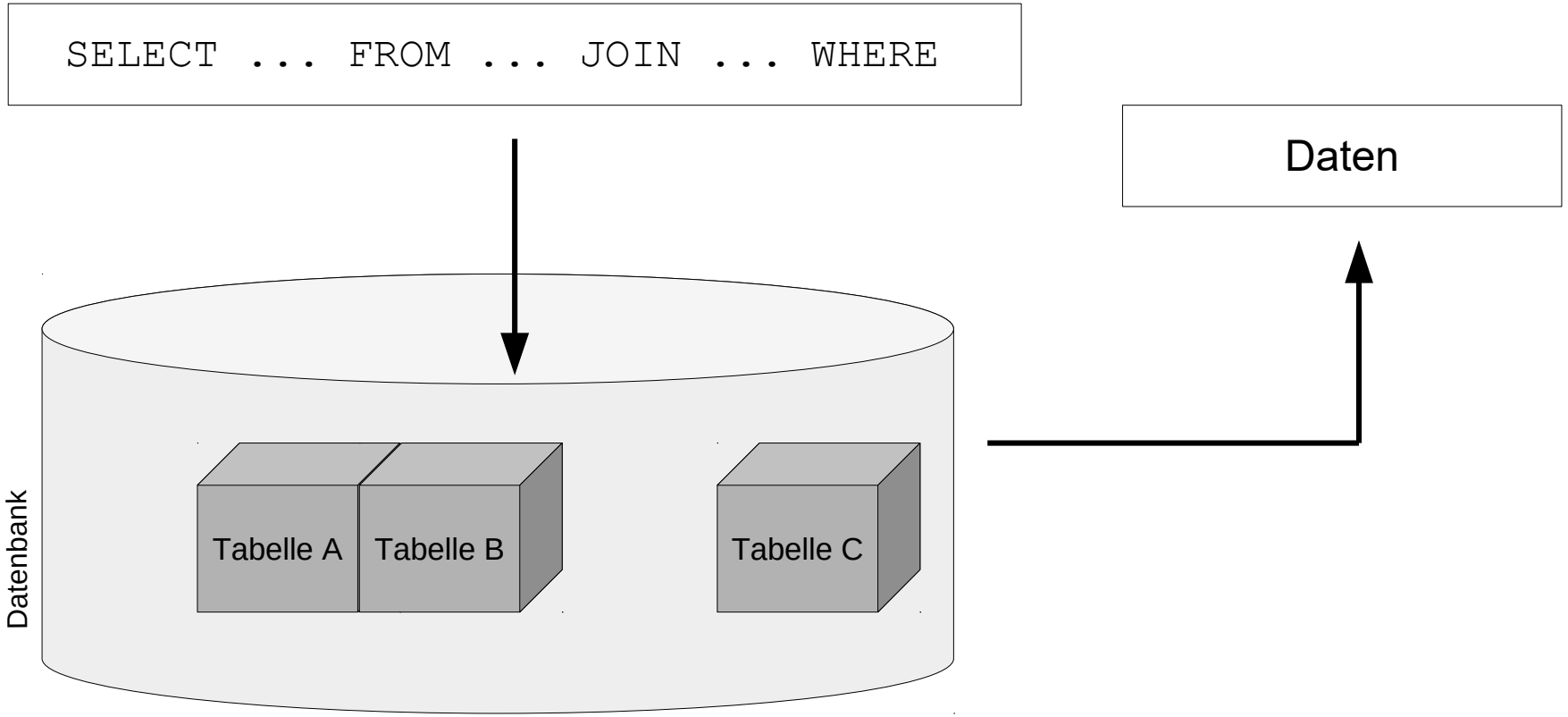


- Eine Beziehung kann Attribute haben.
- Die Beziehung stellt eine eigene „Entität“ dar.
- Die Beziehung wird mit Hilfe von Schlüsselwerten in einer Tabelle plus die Beschreibung abgebildet.
- Häufig bei m:n-Beziehungen.

1 : n - Relation in SQL

```
SELECT
      artists.name,
      albums.Title

FROM artists

INNER JOIN  albums
ON (artists.ArtistId = albums.ArtistId)

ORDER BY albums.Title, artists.Name;
```

Arbeitsweise



Ergebnis einer Auswahlabfrage

- Speicherung in einer temporären Ergebnistabelle.
- Die Ergebnistabelle sammelt die Informationen aus den verschiedenen Datenquellen. Die benötigten Informationen werden in einer Tabelle dargestellt.
- Die Daten in der Ergebnistabelle sind abhängig von den gespeicherten Informationen in den jeweiligen Datenquellen und der Art der Verknüpfung der Quellen.

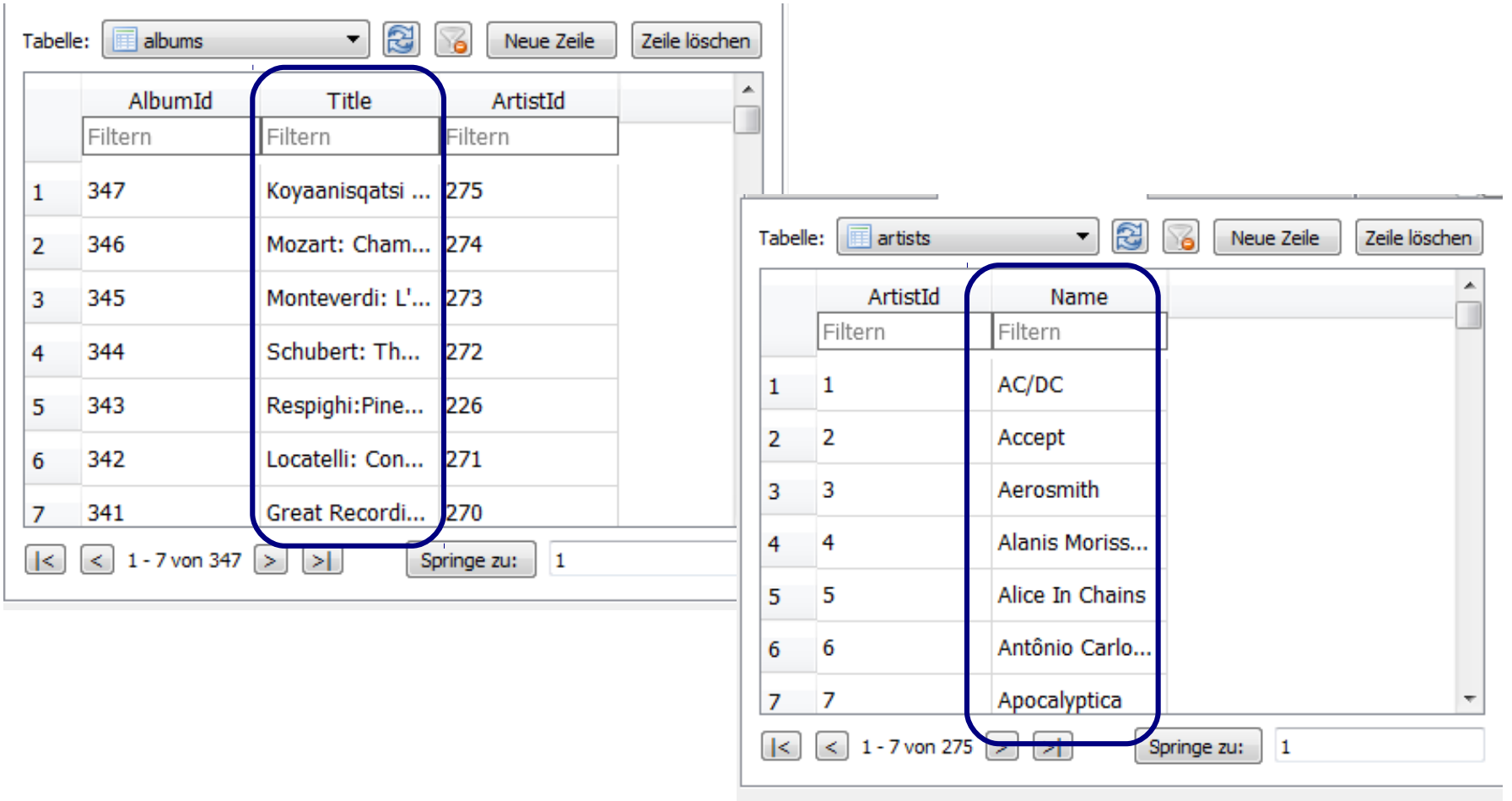
Syntax

```
SELECT [Feld], [Feld]
FROM [Tabelle]

INNER | LEFT OUTER JOIN [Tabelle]
ON [Primärschlüssel] = [Fremdschlüssel]

WHERE ([Bedingung])

ORDER BY [Feld] ASC|DESC, [Feld] ASC|DESC;
```

Auswahl der Datenfelder

```
SELECT
      artists.name,
      albums.Title

FROM artists

INNER JOIN  albums
ON (artists.ArtistId = albums.ArtistId)

ORDER BY albums.Title, artists.Name;
```

… in den Datenquellen



Sortierung der Daten

```
SELECT
      artists.name,
      albums.Title

FROM artists

INNER JOIN  albums
ON (artists.ArtistId = albums.ArtistId)

ORDER BY albums.Title, artists.Name;
```

… in der Datenquelle

- Häufig sind die Informationen in der Datenquelle unsortiert.
- Informationen in den Datenquellen werden häufig vom ersten bis zum letzten eingegebenen Datensatz „sortiert“.

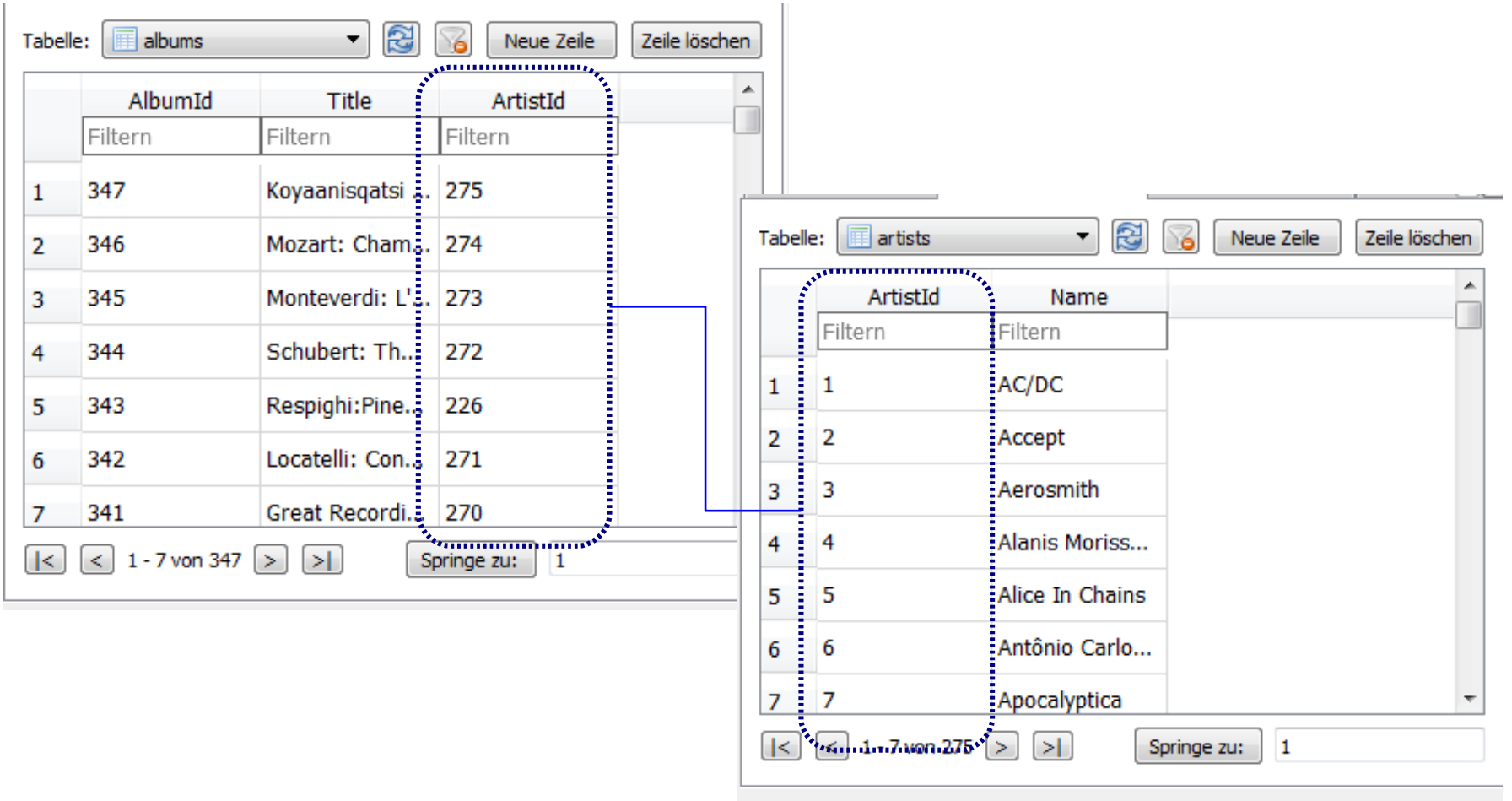
Angabe der Datenquellen

```
SELECT
      artists.name,
      albums.Title

FROM artists
INNER JOIN  albums
ON (artists.ArtistId = albums.ArtistId)

ORDER BY albums.Title, artists.Name;
```

Relationen (Beziehung)

- Verknüpfung von Datensätzen aus zwei Tabellen.
- Verbindung zwischen Master- und Detail-Tabelle.
- Hierarchische Abbildung von Tabellen.
- Die Relation wird in einer SQL-Anweisung mit Hilfe von `FROM … JOIN` abgebildet. Die beiden Tabellen werden unter der Bedingung `ON …` verknüpft.

Verknüpfung mit Hilfe von Schlüsselwerten



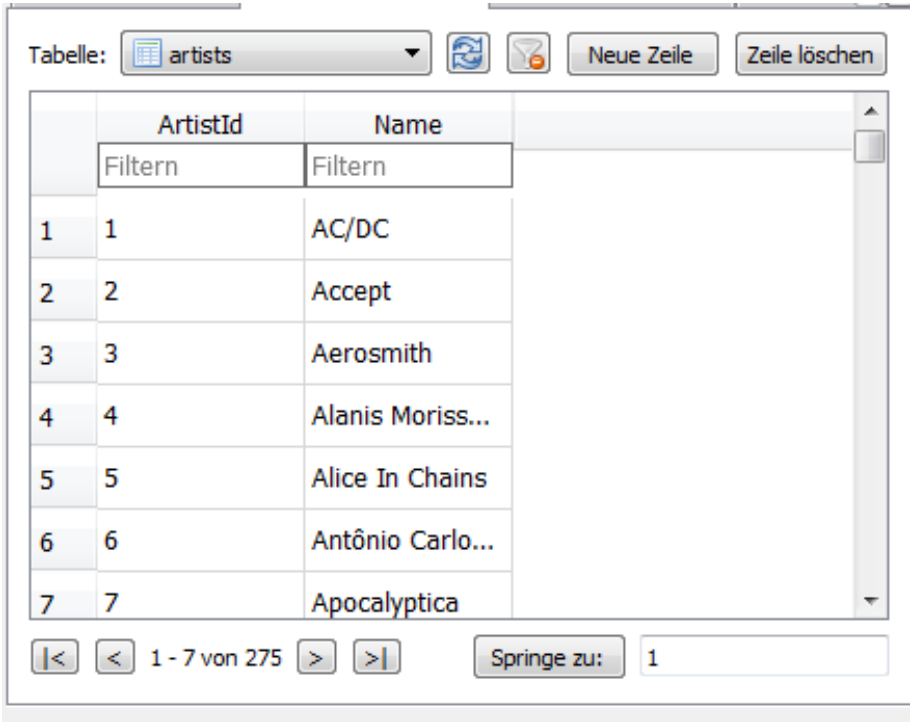
Schlüssel für einen Datensatz

- Jeder Datensatz in einer Tabelle kann mit Hilfe einer ID eindeutig identifiziert werden.
- Jeder Datensatz unterscheidet sich von allen anderen in der Tabelle durch den Schlüsselwert. Der Schlüssel ist eindeutig.
- In Datenbanken werden häufig künstliche Schlüssel genutzt.
- Schlüssel sind zum Beispiel: Artikelnummer, Kundennummer.

Master-Tabelle, rechts vom Join

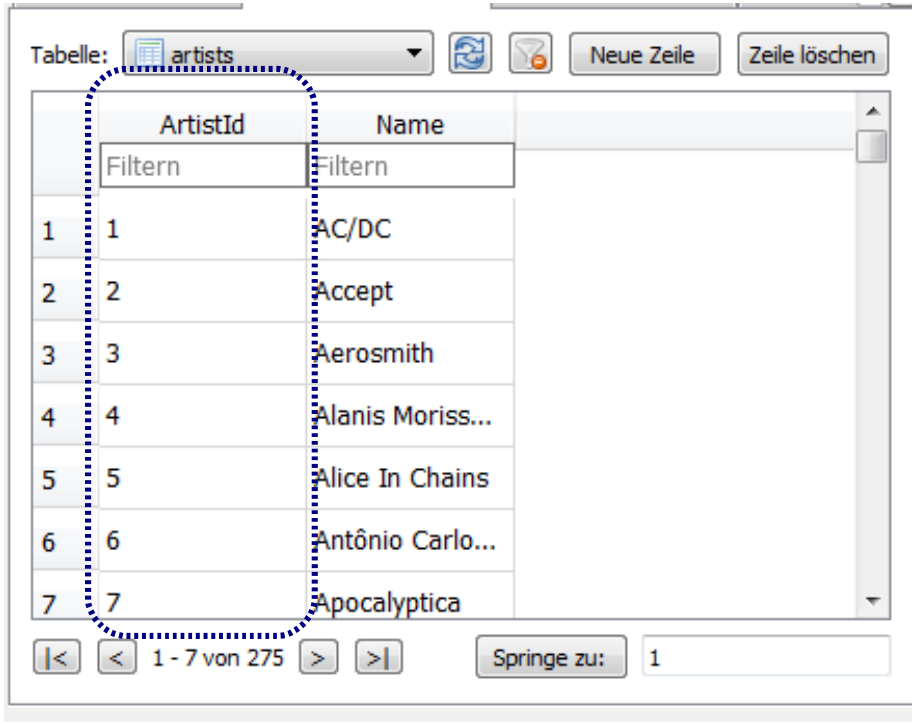
```
SELECT
      artists.name,
      albums.Title

FROM artists
INNER JOIN  albums
ON (artists.ArtistId = albums.ArtistId)

ORDER BY albums.Title, artists.Name;
```

Master-Tabelle (Primärtabelle)



- Abbildung der Hauptdatensätze.
- Jeder Datensatz wird durch einen Primärschlüssel identifiziert.
- Die oberste Tabelle in der Hierarchie enthält keine Schlüsselwerte, die auf andere Tabellen verweisen.

Primärschlüssel in einer Tabelle

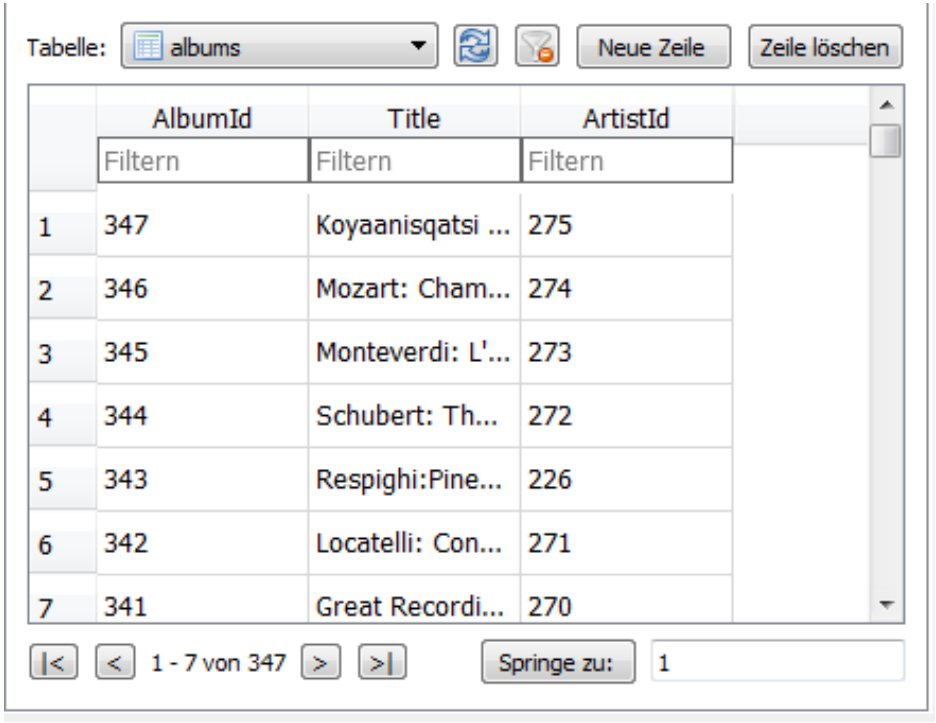


- Eindeutige Identifizierung eines Datensatzes in einer Tabelle.
- Der Schlüsselwert wird sofort bei der Anlage des Datensatzes vergeben.
- Während der Existenz des Datensatzes wird der Schlüssel niemals geändert.

Detail-Tabelle, links vom Join

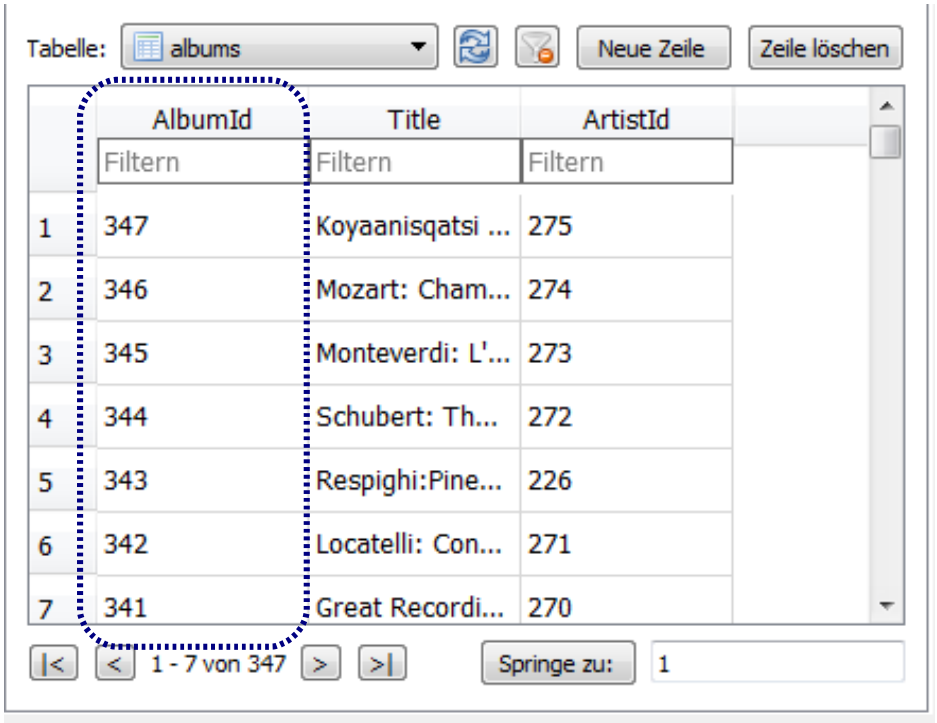
```
SELECT
      artists.name,
      albums.Title

FROM artists
INNER JOIN  albums
ON (artists.ArtistId = albums.ArtistId)

ORDER BY albums.Title, artists.Name;
```

Detail-Tabelle



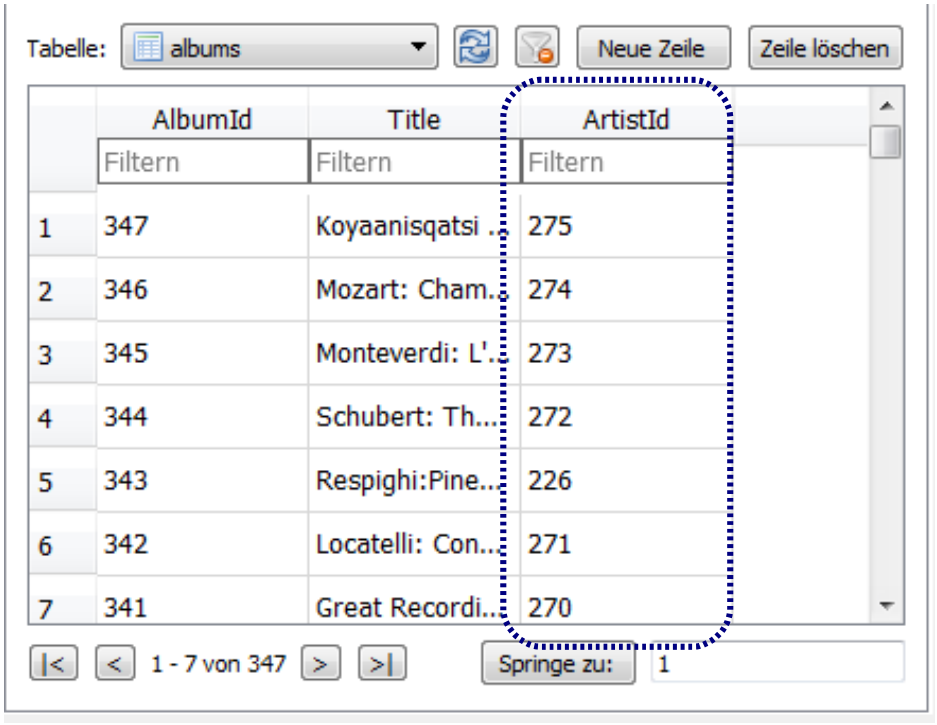
- Abbildung der Detail-Datensätze.
- Nutzung eines Fremdschlüssel, um auf eine andere Tabelle zu verweisen.
- Eine Detail-Tabelle ist das Kind einer Master-Tabelle.

Primärschlüssel



- Jede Detail-Tabelle hat einen Primärschlüssel, der einen Datensatz in dieser Tabelle eindeutig identifiziert.
- Eine Detail-Tabelle kann auch wieder Master-Tabelle sein. Der Primärschlüssel dieser Tabelle wird in einer anderen untergeordneten Tabelle genutzt.

Fremdschlüssel



- Verweis auf einen Datensatz in einer Master-Tabelle.
- Der Primärschlüssel der Master-Tabelle kommt beliebig oft als Fremdschlüssel in einer Detail-Tabelle vor.
- Primär- und Fremdschlüssel sollten vom gleichen Datentyp sein.

Verknüpfung von Primär- und Fremdschlüssel

```
SELECT
      artists.name,
      albums.Title

FROM artists
INNER JOIN  albums
ON (artists.ArtistId = albums.ArtistId)

ORDER BY albums.Title, artists.Name;
```

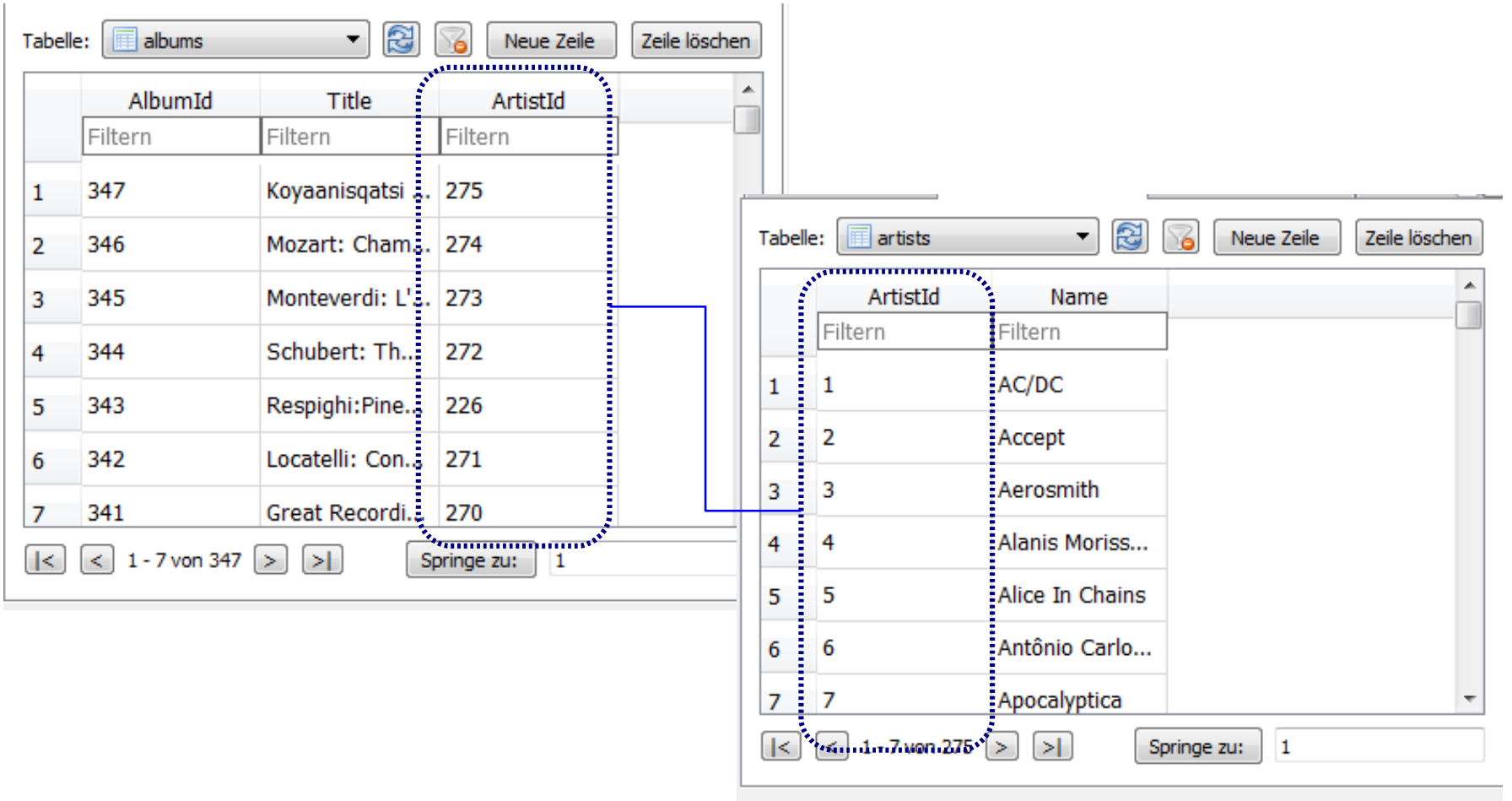
Erläuterung

- Die SQL-Anweisung `ON` legt die Bedingung für die Verknüpfung fest. Mit welchen Feldern wird die Verknüpfung ausgeführt?
- Der Wert in der Spalte „Fremdschlüssel“ in der Detail-Tabelle ist gleich einem Wert in der Spalte „Primärschlüssel“ der dazugehörigen Master-Tabelle.
- Primär- und Fremdschlüssel sollten vom gleichen Datentyp sein.

Hinweis

```
SELECT
      artists.name,
      albums.Title

FROM artists
INNER JOIN  albums
ON (artists.ArtistId = albums.ArtistId)

ORDER BY albums.Title, artists.Name;
```

Abbildung in den Datenquellen



INNER JOIN - Verknüpfung

```
SELECT
      artists.name,
      albums.Title

FROM artists
INNER JOIN  albums
ON (artists.ArtistId = albums.ArtistId)

ORDER BY albums.Title, artists.Name;
```

Regeln für die Anzeige

- Zu einem Fremdschlüssel in der Detail-Tabelle muss ein Primärschlüssel in der Master-Tabelle vorhanden sein.
- Jeder Primärschlüssel aus der Master-Tabelle muss mindestens einmal als Fremdschlüssel in der Detail-Tabelle verwendet werden.

Beispiel

Primär-schlüssel	Erdteil
1	Europa
2	Asien
3	Afrika
4	Amerika

Primär-schlüssel	Fremd-schlüssel	Land
1	1	Belgien
2	1	Polen
3	3	Algier
4	2	Indien
5		Spanien

InnerJoin

Erdteil	Land
Europa	Belgien
Europa	Polen
Afrika	Algier
Asien	Indien

OUTER JOIN - Verknüpfung

```
SELECT
      artists.name,
      albums.Title

FROM artists
LEFT OUTER JOIN  albums
ON (artists.ArtistId = albums.ArtistId)

ORDER BY albums.Title, artists.Name;
```

Erläuterung

- In einen der beiden Tabellen fehlt ein passender Wert.
- Die Tabelle links (LEFT) oder rechts (RIGHT) vom `JOIN` wird vollständig angezeigt. Aus dieser Tabelle werden alle Datensätze in die Ergebnistabelle übernommen.
- Aus der Detail-Tabelle werden nur die passenden Informationen angezeigt. In allen anderen Fällen ist die Information in der Ergebnistabelle nicht vorhanden. Die Felder haben den Wert Null.

LEFT OUTER JOIN - Verknüpfung

```
SELECT
      artists.name,
      albums.Title

FROM artists
LEFT OUTER JOIN  albums
ON (artists.ArtistId = albums.ArtistId)

ORDER BY albums.Title, artists.Name;
```

Beispiel

Primär-schlüssel	Erdteil
1	Europa
2	Asien
3	Afrika
4	Amerika

Primär-schlüssel	Fremd-schlüssel	Land
1	1	Belgien
2	1	Polen
3	3	Algier
4	2	Indien
5		Spanien

Left Outer Join

Erdteil	Land
Europa	Belgien
Europa	Polen
Afrika	Algier
Asien	Indien
Amerika	

RIGHT OUTER JOIN - Verknüpfung

```
SELECT
      artists.name,
      albums.Title

FROM artists
RIGHT OUTER JOIN  albums
ON (artists.ArtistId = albums.ArtistId)

ORDER BY albums.Title, artists.Name;
```

Beispiel

Primär-schlüssel	Erdteil
1	Europa
2	Asien
3	Afrika
4	Amerika

Primär-schlüssel	Fremd-schlüssel	Land
1	1	Belgien
2	1	Polen
3	3	Algier
4	2	Indien
5		Spanien

Right Outer Join

Erdteil	Land
Europa	Belgien
Europa	Polen
Afrika	Algier
Asien	Indien
	Spanien

Hinweise

- Eine RIGHT OUTER JOIN – Verknüpfung ist nicht in SQLite implementiert.
- Durch Tausch der Tabellennamen kann eine RIGHT OUTER JOIN – in eine LEFT OUTER JOIN – Verknüpfung umgewandelt werden.

Kombination von JOIN

```
SELECT
      artists.name,
      albums.Title

FROM tracks
INNER JOIN (albums
      LEFT OUTER JOIN  artists
            ON (albums.ArtistId = artists.ArtistId))
ON (tracks.AlbumId = albums.AlbumId)

ORDER BY albums.Title, artists.Name;
```

Erläuterung

- Entsprechend der Klammerung werden die Verknüpfungen von innen nach außen aufgelöst.
- Einer OUTER JOIN – Verknüpfung sollte keine INNER JOIN – Verknüpfung folgen.

Verknüpfung; gesamt

```
FROM playlists

INNER JOIN (playlist_track

      INNER JOIN tracks
      On (playlist_track.TrackId = tracks.TrackId))

ON (playlist_track.PlaylistId = playlists.PlaylistId)
```

- Die Schlüsselwerte in der Tabelle `playlist_track` werden durch die entsprechenden Informationen aus den Tabellen `playlists` und `tracks` ersetzt.

m : n - Relation

```
SELECT playlists.Name, tracks.Name

FROM playlists

INNER JOIN (playlist_track
      INNER JOIN tracks
      On (playlist_track.TrackId = tracks.TrackId))
ON (playlist_track.PlaylistId = playlists.PlaylistId)

ORDER BY playlists.Name;
```

Innere Verknüpfung

```
FROM playlist_track INNER JOIN tracks
On (playlist_track.TrackId = tracks.TrackId))
```

- Die Quelle `playlist_track` ist die Master-Tabelle. Daten aus dieser Tabelle werden nur angezeigt, wenn der dazugehörige Primärschlüssel mindestens einmal in der Quelle `tracks` vorkommt.
- Die, in der Detail-Tabelle `tracks` genutzten Fremdschlüssel müssen in der Master-Tabelle definiert sein

Beispiel

Primär-schlüssel	TrackID
t1	Track A
t2	Track B
t3	Track C
t4	Track D

Primär-schlüssel	TrackID	
1	t1	
2	t2	
3	t4	
4	t1	

InnerJoin

Track	
Track A	
Track B	
Track D	

Äußere Verknüpfung

```
FROM playlists
INNER JOIN (playlist_track
ON (playlist_track.PlaylistId = playlists.PlaylistId)
```

- Die Quelle `playlists` ist die Master-Tabelle. Daten aus dieser Tabelle werden nur angezeigt, wenn der dazugehörige Primärschlüssel mindestens einmal in der Quelle `playlist_tracks` vorkommt.
- Die, in der Detail-Tabelle `playlist_tracks` genutzten Fremdschlüssel müssen in der Master-Tabelle definiert sein

Beispiel

Primär-schlüssel	PlaylistID
p1	List A
p2	List B
p3	List C
p4	List D

Primär-schlüssel	TrackID	PlaylistID
1		p1
2		p3
3		p1
4		p2

InnerJoin

Track	Playlist
	List A
	List C
	List B

Verknüpfung; gesamt

```
FROM playlists

INNER JOIN (playlist_track

      INNER JOIN tracks
      On (playlist_track.TrackId = tracks.TrackId))

ON (playlist_track.PlaylistId = playlists.PlaylistId)
```

- Die Schlüsselwerte in der Tabelle `playlist_track` werden durch die entsprechenden Informationen aus den Tabellen `playlists` und `tracks` ersetzt.

Äußere und innere Verknüpfung

Primär-schlüssel	TrackID
t1	Track A
t2	Track B
t3	Track C
t4	Track D

Primär-schlüssel	PlaylistID
p1	List A
p2	List B
p3	List C
p4	List D

Primär-schlüssel	TrackID	PlaylistID
1	t1	p1
2	t2	p3
3	t4	p1
4	t1	p2

InnerJoin

Track	Playlist
Track A	List A
Track B	List C
Track D	List A
Track A	List B

Inkonsistenzsuche

```
SELECT albums.Title, albums.ArtistId
FROM albums LEFT OUTER JOIN artists
ON albums.ArtistId = artists.ArtistId
WHERE albums.ArtistId is NULL
ORDER BY albums.Title;
```

- Ist der verwendete Fremdschlüssel in der Detail-Tabelle in der Master-Tabelle vorhanden?
- In diesem Beispiel wird untersucht, ob die Angaben der Kategorie in der Detail-Tabelle `tracks` in der Master-Tablle `genres` definiert sind.

Nicht-Verwendung einer ID aus dem Master

```
SELECT artists.Name, albums.ArtistId
FROM artists LEFT OUTER JOIN albums
ON artists.ArtistId = albums.ArtistId
WHERE albums.ArtistId is NULL
ORDER BY artists.Name;
```

- Wird der Primärschlüssel aus der Master-Tabelle in der Detail-Tabelle genutzt?
- In diesem Beispiel wird untersucht, ob Künstler vorhanden sind, die kein Album erzeugt haben.