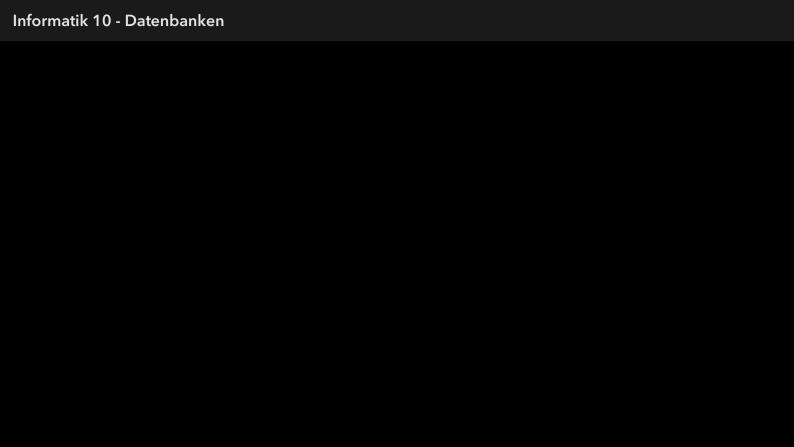
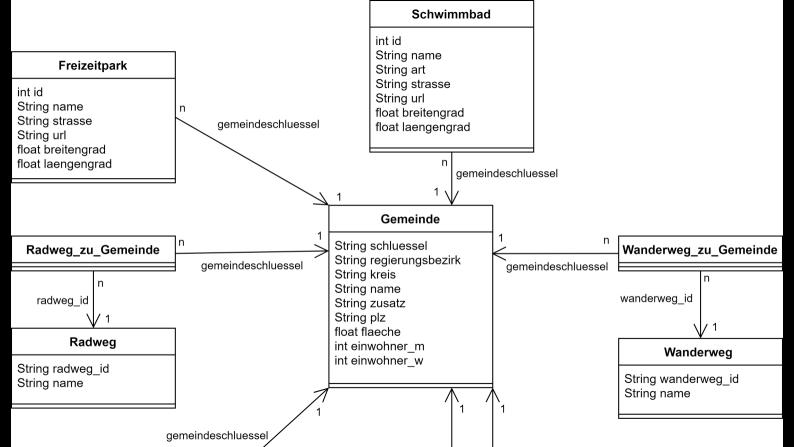


Informatik 10 - Datenbanken







Verändere die SQL-Abfrage so, dass die der Namen und Straßen aller Freizeitparks und die Namen der jeweils zugehörigen Gemeinde ausgegeben wird.

SELECT Freizeitpark.name, Gemeinde.name

FROM Freizeitpark, Gemeinde

Schreibe eine SQL-Abfrage, die Namen und Art aller Schwimmbäder und den Namen und alle Einwohnerzahlen der zugehörigen Gemeinden ausgibt.

Schreibe eine SQL-Abfrage, die die Anzahl an Schwimmbädern in Gemeinden mit **mehr** als 1000 weiblichen Einwohnerinnen ausgibt.

Tipp: Hier brauchst du mehrere verknüpfte Bedingungen

Schreibe eine SQL-Abfrage, die die Namen aller Gemeinde in Oberbayern oder Niederbayern, zu denen ein Wanderweg führt, ausgibt. Dopplungen dürfen auftreten und sollte nicht entfernt werden!

Tipp: Hier brauchst du wieder mehrere verknüpfte Bedingungen. Überlege bei der Verknüpfung von Bedingungen, ob du Klammern setzen musst!

Schreibe eine SQL-Abfrage, die aus den Tabellen Gemeinde und Wanderweg_zu_Gemeinde die Anzahl der Wanderwege, die zu Gemeinden mit mehr als 500 000 männlichen Einwohnern führen, ausgibt.

Schreibe eine SQL-Abfrage, die eine Liste mit den Namen aller Gemeinden, die ein Freibad"haben, und die Namen der jeweiligen Freibäder ausgibt.

AND Schwimmbad.art=Freibad

SELECT COUNT(*)
FROM Gemeinde, Radweg_zu_Gemeinde
WHERE Gemeinde.schluessel=Radweg_zu_Gemeinde.gemeindeschluessel
AND Gemeinde.plz > 96400

SELECT Zoo.name FROM Zoo,Gemeinde WHERE Zoo.gemeindeschluessel = Gemeinde.schluessel AND Gemeinde.name=Ërlangen

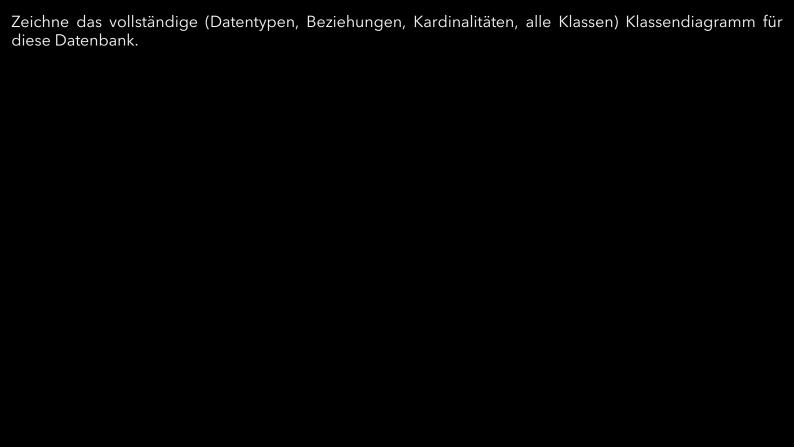
SELECT Radweg_zu_Gemeinde.radweg_id
FROM Radweg_zu_Gemeinde, Gemeinde
WHERE Gemeinde.schluessel = Radweg_zu_Gemeinde.gemeindeschluessel
AND (Gemeinde.regierungsbezirk = Öberfranken"

SELECT Zoo.name
FROM Zoo,Gemeinde
WHERE Zoo.gemeindeschluessel = Gemeinde.schluessel

OR Gemeinde.regierungsbezirk=Ünterfranken")

AND Gemeinde.name=Ërlangen

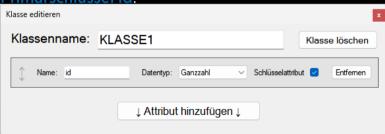
Schreibe eine SQL-Abfrage, die die Anzahl an Radwegen, die an Gemeinden im PLZ-Bereich größer als 96400 angrenzen, ausgibt.



- Kopiere aus dem Vorlagenordner des Ressourcen-Laufwerks (R:/gy0187/klassen/10x/Vorlagen/) die Datei YoungDB.exe in dein Laufwerk H:/ (alternativ als Download: klassenkarte.de/index.php/tools/youngdb/)
- 2. Öffne nun das Programm YoungDB mit einem Doppelklick auf YoungDB.exe.
- 3. Lege ein neues Datenbankmodell an und speichere es auf deinem Laufwerk H:/. Dieses kann mit einem Klick auf **Modell speichern unter** gespeichert werden und mit **Modell laden** wieder geöffnet werden.

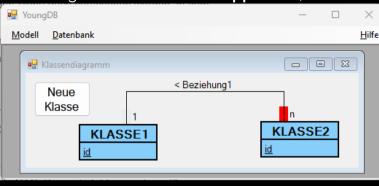


4. Lege zwei neue Klassen an, bearbeite sie mit **Doppelklick** und erstelle jeweils einen ganzzahligen Primärschlüssel id



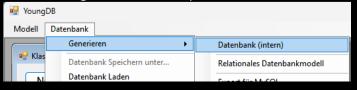
5. Erstelle eine Beziehung zwischen den beiden Klassen, indem du mit **Rechtsklick, Halten und Ziehen** eine rote Linie aus einer der Klassen zur anderen ziehst und dann die Maus loslässt. Bearbeite die

Beziehung anschließend mit **Doppelklick**, sodass sie eine 1:n Beziehung von Klasse2 zu Klasse1 ist.



Tipp: Klassen kannst du per Doppelklick und **Klasse löschen** entfernen und eine Beziehung, indem du den roten Kasten, der erscheint, wenn deine Maus am Anfang der Linie ist, irgendwo hin ziehst.

- 6. Beantworte: Welche Unterschiede stellst du zu normalen Klassendiagrammen fest?
- 7. Erstelle aus dem Modell eine **neue Datenbank und speichere sie** auf deinem Laufwerk H:/. Verwende einen ähnlichen Namen wie für das Modell.Hier kannst du Tabellen öffnen, Daten eintragen, SQL-Abfrage schreiben, speichern und laden.



- 1. Erstelle in YoungDB ein Datenbankmodell mit den Klassen Lehrkraft und Schulklasse, die in einer m:n-Beziehung zueinander stehen. Überlege dir sinnvolle Primärschlüssel und 1-2 Attribute und eine sinnvolle Bezeichnung für die Beziehung.
- 2. Generiere nun die zugehörige Datenbank und befülle die Tabellen mit jeweils 2-3 Datensätzen.
- 3. Beantworte folgende Fragen:
- Welchen essentiellen Unterschied gibt es zwischen m:n-Beziehungen und 1:n-/1:1-Beziehungen?
 - Was für Datensätze werden in der dritten Tabelle eingetragen?
 - Welche Spalten welcher Tabelle(n) sind Fremdschlüssel?
 - Welche Spalte(n) sind Primärschlüssel in der dritten Tabelle? Tipp: Was muss eindeutig sein? Probiere deine Vermutung aus, indem du versucht mehrere Datensätze mit gleichem (vermuteten) Primärschlüssel einzufügen.
 - o Ist es sinnvoll bei m:n-Beziehungen im Klassendiagramm eine Richtung anzugeben und wieso?
 - o Wie könnte man m:n-Beziehung im Klassendiagramm alternativ darstellen?
 - Zeichne die Darstellung von m:n-Beziehungen in YoungDB ein:

LEHRKRAFT kuerzel

SCHULKLASSE bezeichner

Zeichne die Klassenkarten der Tabellen und Song und Playlist. Zeichne anschließend mit zwei verschiedenen Farben die beiden Darstellungsmöglichkeiten der Beziehung zwischen den beiden Tabellen ein.

Bearbeite dann folgende SQL-Aufgaben auf der Website www.dbiu.de/songs und notiere die
getesteten Abfragen.
1 Walaha Sanga (alla Attributa) sind in irgandainar Playlist anthaltan?

- weiche Songs (alle Attribute) sind in Irgendeiner Playlist enthalten?
- 2. Gib die Titel aller Playlists und die Titel der jeweils zugehörigen Songs aus.

3. Welche Songs (alle Attribute) sind in der Playlist namens 'Fussballhits' enthalten?

Tabellenbeziehungen: Fremdschlüssel

Wenn Datensätze mittels Primärschlüssel in einer anderen Tabelle verwendet

werden, spricht man dort von einem Fremdschlüssel. Im Tabellenschema werden die durch () (manchmal auch

) markiert. Ein Beispiel in SQL-Island ist der Häuptling eines Dorfes, der in der Tabelle

Dorf mittels bewohnernr eingetragen wird. Die bewohnernr ist hierbei in der

Wenn Datensätze mittels Primärschlüssel in einer anderen Tabelle verwendet werden, spricht man dort von einem Fremdschlüssel. Im Tabellenschema werden

die **Fremdschlüssel** durch () (manchmal auch) markiert. Ein Beispiel in SQL-Island ist der Häuptling eines Dorfes, der in der Tabelle

Dorf mittels bewohnernr eingetragen wird. Die bewohnernr ist hierbei in der

Wenn Datensätze mittels Primärschlüssel in einer anderen Tabelle verwendet werden, spricht man dort von einem Fremdschlüssel. Im Tabellenschema werden

die **Fremdschlüssel** durch () (manchmal auch) markiert. Ein Beispiel in SQL-Island ist der Häuptling eines Dorfes, der in der Tabelle

Dorf mittels bewohnernr eingetragen wird. Die bewohnernr ist hierbei in der

Wenn Datensätze mittels Primärschlüssel in einer anderen Tabelle verwendet werden, spricht man dort von einem Fremdschlüssel. Im Tabellenschema werden

die **Fremdschlüssel** durch () (manchmal auch) markiert. Ein Beispiel in SQL-Island ist der Häuptling eines Dorfes, der in der Tabelle

Dorf mittels bewohnernr eingetragen wird. Die bewohnernr ist hierbei in der



Die Kardinalität beschreibt, wie viele Objekte auf jeder Seite einer Beziehung stehen können. Es gibt folgende Arten:

- 1:1, z.B. Häuptling pro Dorf, der auch nur in einem Dorf Häuptling ist.
- 1:n, z.B. jeder Bewohner wohnt in einem Dorf, das aber Bewohner hat.
- m:n, z.B. Lehrer pro Schulklasse + Schulklassen pro Lehrer (in

Die Kardinalität beschreibt, wie viele Objekte auf jeder Seite einer Beziehung stehen können. Es gibt folgende Arten:

- 1:1, z.B. ein Häuptling pro Dorf, der auch nur in einem Dorf Häuptling ist.
- 1:n, z.B. jeder Bewohner wohnt in einem Dorf, das aber Bewohner hat.
- m:n, z.B. Lehrer pro Schulklasse + Schulklassen pro Lehrer (in

• m:n, z.B.



Schulklassen pro Lehrer (in

• 1:1, z.B. ein Häuptling pro Dorf, der auch nur in einem Dorf Häuptling ist.

Lehrer pro Schulklasse +

- 1:n, z.B. jeder Bewohner wohnt in einem Dorf, das aber mehrere Bewohner hat.



- 1:1, z.B. ein Häuptling pro Dorf, der auch nur in einem Dorf Häuptling ist.
- 1:n, z.B. jeder Bewohner wohnt in einem Dorf, das aber mehrere Bewohner hat.
- m:n, z.B. mehrere Lehrer pro Schulklasse + Schulklassen pro Lehrer (in



Die Kardinalität beschreibt, wie viele Objekte auf jeder Seite einer Beziehung stehen können. Es gibt folgende Arten:

- 1:1, z.B. ein Häuptling pro Dorf, der auch nur in einem Dorf Häuptling ist.
- 1:n, z.B. jeder Bewohner wohnt in einem Dorf, das aber mehrere Bewohner hat.
- m:n, z.B. mehrere Lehrer pro Schulklasse + mehrere Schulklassen pro Lehrer (in



Möchte man Daten aus zwei Tabellen mit Beziehung zueinander abfragen, gibt man beide Tabellen mit Komma getrennt nach FROM an.

Die SQL-Abfrage bildet dann das der Tabellen. Die Ergebnistabelle enthält von Datensätzen beider Tabellen (Merkregel:).

Um nur zusammengehörige Datensätze (also solche, die miteinenader in Beziehung stehen, z.B. eine Bewohner mit seinem Dorf) auszuwählen, ergänzt man als **Selektion** eine **Gleichheitsbedingung** zwischen Fremd- und zugehörigem . Dann spricht man von einem .

Zum Beispiel kann man in SQL-Island die Daten aller Dörfer und ihrer zugehörigen Häuptlinge so ausgeben:



Möchte man Daten aus zwei Tabellen mit Beziehung zueinander abfragen, gibt man beide Tabellen mit Komma getrennt nach FROM an.

Die SQL-Abfrage bildet dann das **Kreuzprodukt** der Tabellen. Die Ergebnistabelle enthält von Datensätzen beider Tabellen (Merkregel:).

Um nur zusammengehörige Datensätze (also solche, die miteinenader in Beziehung stehen, z.B. eine Bewohner mit seinem Dorf) auszuwählen, ergänzt man als **Selektion** eine **Gleichheitsbedingung** zwischen Fremd- und zugehörigem . Dann spricht man von einem .

Zum Beispiel kann man in SQL-Island die Daten aller Dörfer und ihrer zugehörigen Häuptlinge so ausgeben:



Möchte man Daten aus zwei Tabellen mit Beziehung zueinander abfragen, gibt man beide Tabellen mit Komma getrennt nach FROM an.

Die SQL-Abfrage bildet dann das **Kreuzprodukt** der Tabellen. Die Ergebnistabelle enthält **all Kombinationen** von Datensätzen beider Tabellen (**Merkregel**:).

Um nur zusammengehörige Datensätze (also solche, die miteinenader in Beziehung stehen, z.B. eine Bewohner mit seinem Dorf) auszuwählen, ergänzt man als **Selektion** eine **Gleichheitsbedingung** zwischen Fremd- und zugehörigem . Dann spricht man von einem .

Zum Beispiel kann man in SQL-Island die Daten aller Dörfer und ihrer zugehörigen Häuptlinge so ausgeben:



Möchte man Daten aus zwei Tabellen mit Beziehung zueinander abfragen, gibt man beide Tabellen mit Komma getrennt nach FROM an.

Die SQL-Abfrage bildet dann das Kreuzprodukt der Tabellen. Die Ergebnistabelle enthält alle Kombinationen von Datensätzen beider Tabellen (Merkregel: Jeder mit Jedem).

Um nur zusammengehörige Datensätze (also solche, die miteinenader in Beziehung stehen, z.B. eine Bewohner mit seinem Dorf) auszuwählen, ergänzt man als **Selektion** eine **Gleichheitsbedingung** zwischen Fremd- und zugehörigem . Dann spricht man von einem .

Zum Beispiel kann man in SQL-Island die Daten aller Dörfer und ihrer zugehörigen Häuptlinge so ausgeben:



Möchte man Daten aus zwei Tabellen mit Beziehung zueinander abfragen, gibt man beide Tabellen mit Komma getrennt nach FROM an.

Die SQL-Abfrage bildet dann das Kreuzprodukt der Tabellen. Die Ergebnistabelle enthält alle Kombinationen von Datensätzen beider Tabellen (Merkregel: Jeder mit Jedem).

Um nur zusammengehörige Datensätze (also solche, die miteinenader in Beziehung stehen, z.B. eine Bewohner mit seinem Dorf) auszuwählen, ergänzt man als **Selektion** eine **Gleichheitsbedingung** zwischen Fremd- und zugehörigem **Primärschlüssel**. Dann spricht man von einem .

Zum Beispiel kann man in SQL-Island die Daten aller Dörfer und ihrer zugehörigen Häuptlinge so ausgeben:



Möchte man Daten aus zwei Tabellen mit Beziehung zueinander abfragen, gibt man beide Tabellen mit Komma getrennt nach FROM an.

Die SQL-Abfrage bildet dann das Kreuzprodukt der Tabellen. Die Ergebnistabelle enthält alle Kombinationen von Datensätzen beider Tabellen (Merkregel: Jeder mit Jedem).

Um nur zusammengehörige Datensätze (also solche, die miteinenader in Beziehung stehen, z.B. eine Bewohner mit seinem Dorf) auszuwählen, ergänzt man als **Selektion** eine **Gleichheitsbedingung** zwischen Fremd- und zugehörigem **Primärschlüssel**. Dann spricht man von einem **Join**.

Zum Beispiel kann man in SQL-Island die Daten aller Dörfer und ihrer zugehörigen Häuptlinge so ausgeben:

Aufgaben



Alle Aufgaben beziehen sich auf die Datenbank oben. Eine Online-Version gibt es unter www.dbiu.de/bayern/.

Gib immer genau die geforderten Daten aus und nicht mehr. Sortiere nicht, wenn du nicht dazu aufgefordert wirst.

Verändere die SQL-Abfrage so, dass die Namen und Internetadressen (=url) aller Zoos und der Name und Regierungsbezirk der jeweiligen Gemeinde ausgegeben wird:

SELECT Zoo.name, Gemeinde.name FROM Zoo, Gemeinde

Aufgaben



Alle Aufgaben beziehen sich auf die Datenbank oben. Eine Online-Version gibt es unter www.dbiu.de/bayern/.

Gib immer genau die geforderten Daten aus und nicht mehr. Sortiere nicht, wenn du nicht dazu aufgefordert wirst.

Verändere die SQL-Abfrage so, dass die Namen und Internetadressen (=url) aller Zoos und der Name und Regierungsbezirk der jeweiligen Gemeinde ausgegeben wird:

SELECT Zoo.name, Gemeinde.name ,Gemeinde.regierungsbezirk, Zoo.url FROM Zoo, Gemeinde

Aufgaben



Alle Aufgaben beziehen sich auf die Datenbank oben. Eine Online-Version gibt es unter www.dbiu.de/bayern/.

Gib immer genau die geforderten Daten aus und nicht mehr. Sortiere nicht, wenn du nicht dazu aufgefordert wirst.

Verändere die SQL-Abfrage so, dass die Namen und Internetadressen (=url) aller Zoos und der Name und Regierungsbezirk der jeweiligen Gemeinde ausgegeben wird:

SELECT Zoo.name, Gemeinde.name ,Gemeinde.regierungsbezirk, Zoo.url FROM Zoo, Gemeinde

WHERE Zoo.gemeindeschluessel = Gemeinde.schluessel

YoungDB

m:n-Beziehungen



m:n-Beziehungen können im UML-Klassendiagramm auf zwei verschiedene Arten dargestellt werden:

1.

Vorteil:

Lehrkraft

String kuerzel

Vorteil:

2.

Lehrkraft

String kuerzel

Schulklasse

String bezeichner

Schulklasse



m:n-Beziehungen können im UML-Klassendiagramm auf zwei verschiedene Arten dargestellt werden:

als direkte Beziehung

Vorteil:

Lehrkraft

String kuerzel

2. Vorteil: Schulklasse

String bezeichner

Lehrkraft

String kuerzel

Schulklasse



m:n-Beziehungen können im UML-Klassendiagramm auf zwei verschiedene Arten dargestellt werden:

1. als direkte Beziehung

Vorteil: Diagramm kompakt und übersichtlich

Lehrkraft

String kuerzel

2.

Vorteil:

Schulklasse

String bezeichner

Schulklasse

String bezeichner

String kuerzel

Lehrkraft



m:n-Beziehungen können im UML-Klassendiagramm auf zwei verschiedene Arten dargestellt werden:

1. als direkte Beziehung

Vorteil: Diagramm kompakt und übersichtlich

Lehrkraft

String kuerzel

Schulklasse

String bezeichner

2. mit Beziehungstabelle

Vorteil:

Lehrkraft

String kuerzel

Schulklasse



m:n-Beziehungen können im UML-Klassendiagramm auf zwei verschiedene Arten dargestellt werden:

1. als direkte Beziehung

Vorteil: Diagramm kompakt und übersichtlich

Lehrkraft

String kuerzel

Schulklasse

String bezeichner

2. mit Beziehungstabelle

Vorteil: Diagramm kompakt und übersichtlich

Lehrkraft

String kuerzel

Schulklasse

Um zwei Tabellen, die eine m:n-Beziehung miteinander haben, zu joinen (also ihren Join zu bilden

und in der Ergebnistabelle nur Datensätze zu haben), muss man:

- Daten aus allen Tabellen abfragen (also diese nach auflisten).
- Die einzeln mit den normalen Tabellen joinen. Hierfür benötigt man

Join-Bedingungen, die mit verknüpft werden.

Beispiel:
SELECT Lobrkraft * Schulklasso *

SELECT Lehrkraft.*, Schulklasse.*

FROM Lehrkraft, Schulklasse,

WHERE Lehrer_unterricht_Klasse.lehrer =



Um zwei Tabellen, die eine m:n-Beziehung miteinander haben, zu joinen (also ihren Join zu bilden

und in der Ergebnistabelle nur **zusammengehörende** Datensätze zu haben), muss man:

- Daten aus allen Tabellen abfragen (also diese nach auflisten).
- einzeln mit den normalen Tabellen joinen. Hierfür benötigt man Die

Join-Bedingungen, die mit verknüpft werden.

Beispiel: SELECT Lehrkraft.*, Schulklasse.*

FROM Lehrkraft, Schulklasse,

WHERE Lehrer unterricht Klasse.lehrer =



Um zwei Tabellen, die eine m:n-Beziehung miteinander haben, zu joinen (also ihren Join zu bilden

und in der Ergebnistabelle nur **zusammengehörende** Datensätze zu haben), muss man:

- Daten aus allen **drei** Tabellen abfragen (also diese nach auflisten).
- einzeln mit den normalen Tabellen joinen. Hierfür benötigt man Die

Join-Bedingungen, die mit verknüpft werden.

Beispiel:

SELECT Lehrkraft.*, Schulklasse.*

FROM Lehrkraft, Schulklasse,

WHERE Lehrer unterricht Klasse.lehrer =



Um zwei Tabellen, die eine m:n-Beziehung miteinander haben, zu joinen (also ihren Join zu bilden und in der Ergebnistabelle nur **zusammengehörende** Datensätze zu haben), muss man:

- Daten aus allen drei Tabellen abfragen (also diese nach FROM auflisten).
- einzeln mit den normalen Tabellen joinen. Hierfür benötigt man Die

Join-Bedingungen, die mit verknüpft werden.

Beispiel:

SELECT Lehrkraft.*, Schulklasse.*

FROM Lehrkraft, Schulklasse,

WHERE Lehrer unterricht Klasse.lehrer =

Um zwei Tabellen, die eine m:n-Beziehung miteinander haben, zu joinen (also ihren Join zu bilden und in der Ergebnistabelle nur zusammengehörende Datensätze zu haben), muss man:

- Daten aus allen drei Tabellen abfragen (also diese nach FROM auflisten).
- Die **Beziehungstabelle** einzeln mit den normalen Tabellen joinen. Hierfür benötigt man

Join-Bedingungen, die mit verknüpft werden.

Beispiel: SELECT Lehrkraft.*, Schulklasse.*

SELECT Lenrkraft., Schulklasse.

FROM Lehrkraft, Schulklasse,

WHERE Lehrer_unterricht_Klasse.lehrer =

Um zwei Tabellen, die eine m:n-Beziehung miteinander haben, zu joinen (also ihren Join zu bilden und in der Ergebnistabelle nur **zusammengehörende** Datensätze zu haben), muss man:

- Daten aus allen drei Tabellen abfragen (also diese nach FROM auflisten).
- Beziehungstabelle einzeln mit den normalen Tabellen joinen. Hierfür benötigt man zwei

Join-Bedingungen, die mit verknüpft werden.

Beispiel:

SELECT Lehrkraft.*, Schulklasse.*

FROM Lehrkraft, Schulklasse,

WHERE Lehrer unterricht Klasse.lehrer = AND Lehrer_unterricht_Klasse.klasse =

Um zwei Tabellen, die eine m:n-Beziehung miteinander haben, zu joinen (also ihren Join zu bilden und in der Ergebnistabelle nur zusammengehörende Datensätze zu haben), muss man:

- Daten aus allen **drei** Tabellen abfragen (also diese nach **FROM** auflisten).
- Die Beziehungstabelle einzeln mit den normalen Tabellen joinen. Hierfür benötigt man zwei

Join-Bedingungen, die mit AND verknüpft werden.

Beispiel: SELECT Lehrkraft.*, Schulklasse.*

FROM Lehrkraft, Schulklasse,

WHERE Lehrer unterricht Klasse.lehrer =

WHERE Lehrer_unterricht_Klasse.lehrer = AND Lehrer_unterricht_Klasse.klasse =

Um zwei Tabellen, die eine m:n-Beziehung miteinander haben, zu joinen (also ihren Join zu bilden und in der Ergebnistabelle nur **zusammengehörende** Datensätze zu haben), muss man:

- Daten aus allen drei Tabellen abfragen (also diese nach FROM auflisten).
- Die Beziehungstabelle einzeln mit den normalen Tabellen joinen. Hierfür benötigt man zwei

Join-Bedingungen, die mit AND verknüpft werden.

Beispiel: **SELECT** Lehrkraft.*, Schulklasse.*

FROM Lehrkraft, Schulklasse, Lehrer_unterricht_Klasse

WHERE Lehrer unterricht Klasse.lehrer =

Um zwei Tabellen, die eine m:n-Beziehung miteinander haben, zu joinen (also ihren Join zu bilden und in der Ergebnistabelle nur zusammengehörende Datensätze zu haben), muss man:

- Daten aus allen drei Tabellen abfragen (also diese nach FROM auflisten).
- Die **Beziehungstabelle** einzeln mit den normalen Tabellen joinen. Hierfür benötigt man **zwei**

Join-Bedingungen, die mit AND verknüpft werden.

Beispiel: SELECT Lehrkraft.*, Schulklasse.*

FROM Lehrkraft, Schulklasse, Lehrer_unterricht_Klasse

WHERE Lehrer_unterricht_Klasse.lehrer = Lehrkraft.kuerzel
AND Lehrer_unterricht_Klasse.klasse =

Um zwei Tabellen, die eine m:n-Beziehung miteinander haben, zu joinen (also ihren Join zu bilden und in der Ergebnistabelle nur zusammengehörende Datensätze zu haben), muss man:

- Daten aus allen drei Tabellen abfragen (also diese nach FROM auflisten).
- Die Beziehungstabelle einzeln mit den normalen Tabellen joinen. Hierfür benötigt man zwei

Join-Bedingungen, die mit AND verknüpft werden.

Beispiel:

SELECT Lehrkraft.*, Schulklasse.*

FROM Lehrkraft, Schulklasse, Lehrer_unterricht_Klasse
WHERE Lehrer unterricht Klasse.lehrer = Lehrkraft.kuerzel

AND Lehrer_unterricht_Klasse.klasse = Schulklasse.bezeichner

Song-Datenbank Diagramme: www.dbi.u.de/songs

