



Datenbanken und SQL

(Woche 4 - Tag 2)

Agenda

Spezielle JOINS

- STRAIGHT_JOIN
- NATURAL [LEFT, RIGHT] JOIN

STRAIGHT JOIN

Definition + Motivation

- Der **STRAIGHT_JOIN** arbeitet (zusammen mit der entsprechenden ON-Bedingung) wie ein INNER JOIN, durchläuft aber (im Rahmen einer äußeren Schleife) alle Datensätze der linken Tabelle, um (mittels einer inneren Schleife) den **einzigsten verknüpfbaren Datensatz** der rechten Tabelle zu suchen.
- Entsprechend kann der STRAIGHT_JOIN nur sinnvoll bei Beziehungstypen genutzt werden, bei denen mindestens eine der beiden Kardinalitäten den Wert **1** hat. Dies ist allerdings stets gegeben, da m:n-Beziehungen ja mittels Hilfstabelle in zwei 1:n Beziehungen aufgelöst werden.
- Aus dem selben Grund ist der Einsatz des STRAIGHT_JOINS daher nur sinnvoll, wenn die Kardinalität der rechten Tabelle den Wert **1** hat.
- Motivation für den Einsatz des STRAIGHT_JOINS ist eine verbesserte **Performance**, da die Existenz eindeutiger Verknüpfungen bedeutet, dass stets mit dem Auffinden des **einzigsten** möglichen Treffers (innerhalb der rechten Tabelle) der jeweilige Such-Prozess abgebrochen werden kann.

Beispiel



SELECT * FROM Person STRAIGHT_JOIN Ort ON Person.Ort_ID = Ort.Ort_ID;

In einer äußeren Schleife wird pro Durchlauf die jeweils „nächste“ Person ausgewählt.

Pro Person startet eine innere Schleife, die den Geburtsort sucht. Sobald dieser gefunden wurde, kann die Suche abgebrochen werden.

Hinweis:

Der STRAIGHT_JOIN ist nur in solchen Fällen notwendig, in dem das Datenbank-Management-System (DBMS) nicht bereits selbstständig (mittels „**Optimizer**“) die optimale Reihenfolge der Tabellen-Abarbeitung vornimmt.

NATURAL JOIN

Definition + Motivation + Kritik

- Der **NATURAL JOIN** arbeitet wie ein INNER JOIN, benötigt allerdings keine ON-Bedingung, wobei für die beiden zu verknüpfenden Tabellen jedoch die folgenden Voraussetzungen erfüllt sein müssen:
 - Die Verknüpfung soll bezüglich der Bedingung **Tabelle_1.Primärschlüssel = Tabelle_2.Fremdschlüssel** stattfinden.
 - Die **Bezeichnung** von Primärschlüssel und Fremdschlüssel muss **identisch** sein.
 - Die Bezeichnungen **aller anderen Attribute** müssen **unterschiedlich** sein.
- Für die Verwendung des NATURAL JOINs spricht ein **geringerer Schreibaufwand**, wie auch die **verbesserte Lesbarkeit** des Codes (insbesondere bei einer größeren Anzahl von zu verknüpfenden Tabellen).
- Gegner des NATURAL JOINs wenden jedoch ein, dass die (oben bereits angesprochenen) Voraussetzungen problematisch seien, da ehemals korrekte Abfragen mit einem NATURAL JOIN zukünftig zu **Fehlern** führen könnten, falls die Attribut-Bezeichnungen der zu verknüpfenden Tabellen geändert wurden.
- Der NATURAL JOIN ist daher das Bayern München der JOINs. Man liebt, oder hasst ihn ;-).

Beispiel

Aufgabenstellung:

Es sollen **Vor-** und **Nachname** aller Kunden ausgegeben werden, die mindestens einmal von der **RocketLogistic AG** beliefert wurden.

Wir betrachten zunächst die bisherige – „schreib-intensive“ – Lösung:

```
SELECT DISTINCT Vorname, Nachname
FROM Kunde INNER JOIN Abrechnung
    ON Kunde.Kunde_ID=Abrechnung.Kunde_ID
    INNER JOIN Abrechnung_Produkt
    ON Abrechnung.Abrechnung_ID=Abrechnung_Produkt.Abrechnung_ID
    INNER JOIN Produkt
    ON Abrechnung_Produkt.Produkt_ID=Produkt.Produkt_id
    INNER JOIN Hersteller
    ON Produkt.Hersteller_ID=Hersteller.Hersteller_ID
    INNER JOIN Spedition
    ON Hersteller.Spedition_ID=Spedition.Spedition_ID
    AND Spedition_Name="RocketLogistic AG";
```


Beispiel

Aufgabenstellung:

Es sollen **Vor-** und **Nachname** aller Kunden ausgegeben werden, die mindestens einmal von der **RocketLogistic AG** beliefert wurden.

Wir betrachten nun die **komfortable** Lösung:

```
SELECT DISTINCT Vorname, Nachname  
FROM Kunde NATURAL JOIN Abrechnung  
      NATURAL JOIN Abrechnung_Produkt  
      NATURAL JOIN Produkt  
      NATURAL JOIN Hersteller  
      NATURAL JOIN Spedition  
WHERE Spedition_Name="RocketLogistic AG";
```

Hinweis

Der NATURAL JOIN besitzt noch eine weitere komfortable Eigenschaft, die wir am folgenden Beispiel erläutern wollen:

Aufgabenstellung:

Für jede Abrechnung soll **Datum** und **Email-Adresse** des Kunden ausgegeben werden.
Allerdings sollen nur Kunden mit einer **ID > 3** berücksichtigt werden.

Wir betrachten erneut zunächst einen **Lösungsversuch** ohne NATURAL JOIN:

```
SELECT Datum, Email  
FROM Kunde INNER JOIN Abrechnung  
ON Kunde.Kunde_ID=Abrechnung.Kunde_ID  
WHERE Kunde_ID>3;
```

MySQL meldet: ?

```
#1052 - Feld 'Kunde_ID' in where clause ist nicht eindeutig
```

Hinweis

Der NATURAL JOIN besitzt noch eine weitere komfortable Eigenschaft, die wir am folgenden Beispiel erläutern wollen:

Aufgabenstellung:

Für jede Abrechnung soll **Datum** und **Email-Adresse** des Kunden ausgegeben werden.
Allerdings sollen nur Kunden mit einer **ID > 3** berücksichtigt werden.

Korrektur:

```
SELECT Datum, Email  
FROM Kunde INNER JOIN Abrechnung  
ON Kunde.Kunde_ID=Abrechnung.Kunde_ID  
WHERE Kunde.Kunde_ID>3;
```

Datum	Email
2021-11-03	myr@xyz.de
2021-10-25	ehahn@xyz.de
2022-02-14	nix@xyz.de

Hinweis

Der NATURAL JOIN besitzt noch eine weitere komfortable Eigenschaft, die wir am folgenden Beispiel erläutern wollen:

Aufgabenstellung:

Für jede Abrechnung soll **Datum** und **Email-Adresse** des Kunden ausgegeben werden.
Allerdings sollen nur Kunden mit einer **ID > 3** berücksichtigt werden.

Diese Korrektur ist beim NATURAL JOIN **nicht notwendig**, da dieser die beiden Kunden-IDs zu einer zusammenfasst:

```
SELECT Datum, Email  
FROM Kunde NATURAL JOIN Abrechnung  
WHERE Kunde_ID>3;
```

Datum	Email
2021-11-03	myr@xyz.de
2021-10-25	ehahn@xyz.de
2022-02-14	nix@xyz.de

Erläuterung zum Hinweis

Beispiel (1)

INNER JOIN zwischen den Tabellen Kunde und Abrechnung:

```
SELECT *  
FROM Kunde INNER JOIN Abrechnung  
ON Kunde.Kunde_ID=Abrechnung.Kunde_ID;
```

Kunde_ID	Vorname	Nachname	Email	Abrechnung_ID	Kunde_ID	Datum
1	Elli	Rot	rot@xyz.de	1	1	2021-05-05
3	Witali	Myrnov	myr@xyz.de	2	3	2021-10-07
2	Vera	Deise	deise@xyz.de	3	2	2021-10-11
2	Witali	Myrnov	myr@xyz.de	4	2	2021-10-16

Beispiel (2)

NATURAL JOIN zwischen den Tabellen Kunde und Abrechnung:

```
SELECT *  
FROM Kunde NATURAL JOIN Abrechnung;
```

Kunde_ID	Vorname	Nachname	Email	Abrechnung_ID	Datum
1	Elli	Rot	rot@xyz.de	1	2021-05-05
3	Witali	Myrnov	myr@xyz.de	2	2021-10-07
2	Vera	Deise	deise@xyz.de	3	2021-10-11
2	Witali	Myrnov	myr@xyz.de	4	2021-10-16

Man sagt: Die Kunde_ID in der Abrechnungstabelle wurde „ausgeblendet“.

NATURAL [LEFT, RIGHT] JOIN

Definition + Motivation

- Die Funktionalität des NATURAL JOINS kann auch beim **NATURAL LEFT JOIN** (und **NATURAL RIGHT JOIN**) genutzt werden. Auch in diesen Fällen müssen allerdings die genannten Voraussetzungen vorliegen.
- Motivation ist erneut der **geringere Schreibaufwand** und die **verbesserte Lesbarkeit** des Codes.
- Zur Demonstration reicht uns aber ein **Beispiel**, das mit dem NATURAL LEFT JOIN arbeitet, da die Verwendung des NATURAL RIGHT JOINS vollkommen identisch verläuft.

Beispiel

Aufgabenstellung:

Es sollen **Vor-** und **Nachname** aller Kunden ausgegeben werden, zusammen mit der **jeweiligen Anzahl** der bestellten **Produkte**.
Es sollen aber **auch Kunden** berücksichtigt werden, die **kein Produkt** bestellt haben.

Lösung mittels NATURAL LEFT JOIN:

```
SELECT Vorname, Nachname, COUNT(Produkt_ID)
FROM Kunde NATURAL LEFT JOIN Abrechnung
      NATURAL LEFT JOIN Abrechnung_Produkt
GROUP BY Vorname, Nachname;
```

Vorname	Nachname	COUNT(Produkt_ID)
Elli	Rot	5
Eva	Hahn	2
Gala	Nieda	0
Peter	Kaufnix	0
Rita	Myrnov	3
Vera	Deise	5
Witali	Myrnov	5

Vielen Dank für Ihre Aufmerksamkeit!

