Daten-Modell und SQL-Funktionen für Join-Typen und Set-Operatoren

Das Dokument zeigt verschiedene Daten-Modelle-Varianten und die Auswirkungen auf die Nutzungs-Möglichkeit der SQL-Sprach-Funktionen auf.
Insbesondere werden behandelt:

- **Daten-Modell-Typen:**
  - Hierarchie: Subset- und Generalisierte Hierarchie
  - Lockere Objekt-Beziehungen
  - De-Normalisierungs-Maßnahmen
  - Trennung von Daten:
    - Aktuelle, Historie und Zukunft
    - Versionierung (Produktiv und Individuell) mit Virtualisierung/Überlagerung

- **SQL-Statement-Funktionalitäten:**
  - Joins: Inner und Outer Joins
  - Set-Operatoren: UNION, INTERSECT und EXCEPT

 Stand: 1.3.2009

DGD-Dienstleistungsgesellschaft für Datenverarbeitung mbH
Im Weingarten 47

D-65201 Wiesbaden Frauenstein
Tel. 0611 / 94 27 30
Fax 0611 / 42 89 43
Email info@dgd-ub.de
Homepage **http://www.dgd-ub.de**

Inhaltsverzeichnis

1 Daten-Modell und SQL-Funktionsauswirkungen

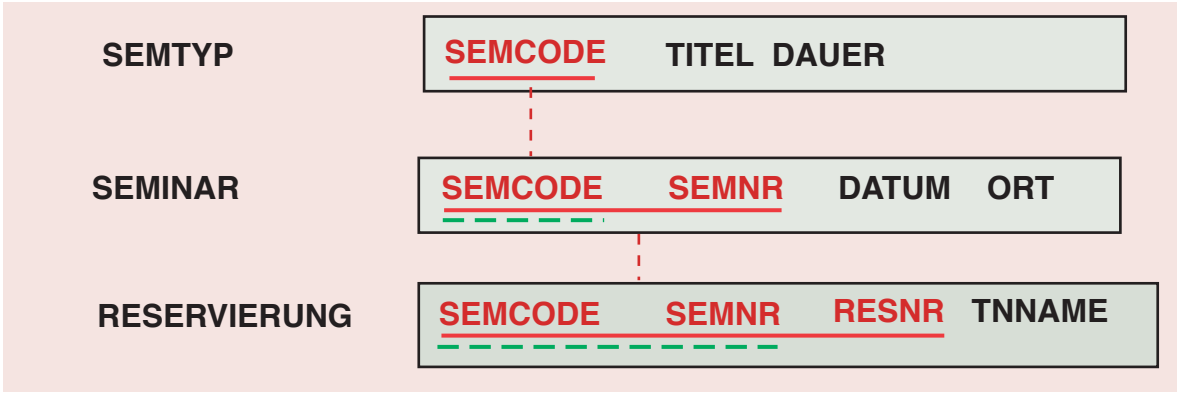
1.1 Join-Typen und Set-Operatoren

1.1.1 Die Einflüsse des logischen konzeptionellen Daten-Modells

1.1.1.1 Modell-Grund-Varianten

Die Grund-Varianten der Struktur-Beziehungen des konzeptionellen Modells

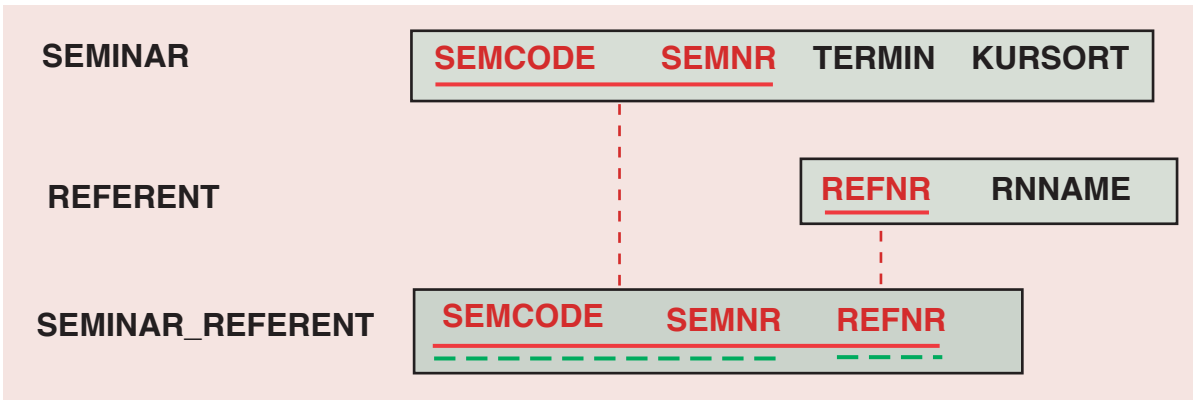
1 Hierarchie



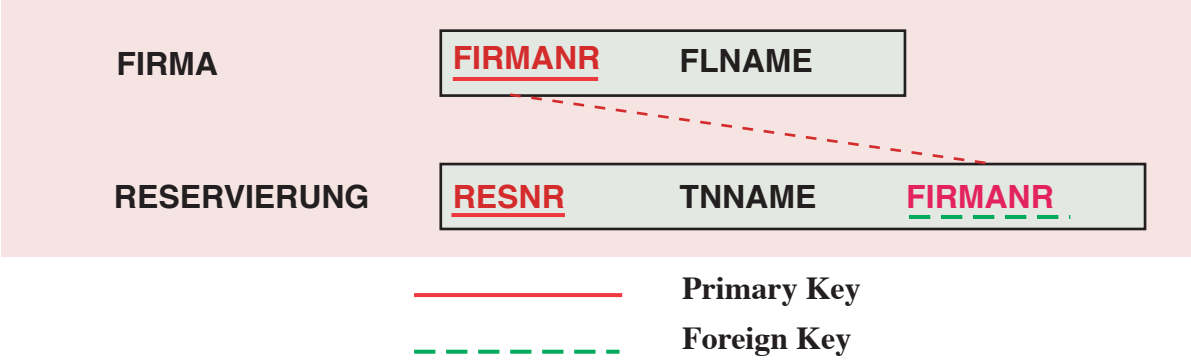
Hinweis:
Die zusammengesetzten PKs sind nicht zwingend und sollten auch i.d.R. nicht genutzt werden. Es liegt auch dann eine Hierarchie vor, wenn jede Tabelle einen PK mit einer einzigen Spalte aufweist:

- SEMINAR: SEMNR
- RESERVIERUNG: RESNR

2 Aufgelöste M : N -Beziehung



3 Lockere Beziehung



Wird das konzeptionelle Daten-Modell nach den Vorschriften des Relationen-Modells entwickelt, besteht es aus den nebenstehend dargestellten drei Grund-Varianten von möglichen Beziehungs-Typen:

- **Hierarchie**
  Bei einer Hierarchie tragen die übergeordneten Objekte die PKs, die untergeordneten Objekte die FKs.
  Die FKs können Teil des PKs sein. In jedem Fall sind sie nicht NULL-fähig.

- **Aufgelöste M : N -Beziehung**
  Bei einer aufgelösten M : N-Beziehung ergibt sich ein Netzwerk mit zwei hierarchisch übergeordneten und einem untergeordneten Objekt.
  Das untergeordnete Objekt trägt zwei FKs, die jeweils auf den übergeordneten PK gerichtet sind.
  Die Regeln entsprechen der vorab dargestellten Hierarchie.
  Die FKs können Teil des PKs sein. In jedem Fall sind sie nicht NULL-fähig.

- **Lockere Beziehung**
  Bei einer lockeren Beziehung stehen zwei Objekte in einer nicht-hierarchischen Beziehung.
  Jedes Objekt verfügt über seinen PK. Aufgrund der technischen Festlegung der Zuordnung des FKs ist anschließend wiederum das grundsätzliche Regelwerk der Hierarchie wirksam; es gilt jedoch die grundsätzliche Ausnahme, dass der FK NULL-fähig sein kann.

Diese Grund-Varianten lassen sich ggf. noch weiter aufgliedern und kategorisieren. Dies gilt insbesondere für die Hierarchie, bei der weitere Spezialfälle erkennbar sind, wie:

- Generalisierte Hierarchie mit Super-Typ und Sub-Typen
- Aggregationen (Kompositionen) mit der Darstellung von Teilen, die mit der Aggregation ein neues Teil bilden.

Außerdem können spezielle Beziehungen definiert werden, wie:

- Rekursive Beziehung eines einzelnen Objekts,
- Rekursive Beziehung von Objekten im Sinne von Zyklen.

Solche Beziehungs-Typen haben Einfluss auf die Ergebnisse der Join-Verarbeitung, da es typischerweise spezielle Vorkommensvarianten von Daten und korrelierenden PKs und FKs gibt.

1.1.2 Die grundsätzlichen Effekte der Join-Typen
1.1.2.1 Join-Verarbeitungs-Regeln

Ein Join entspricht einer Multiplikation.
Die folgende Tabelle zeigt die Wirkungen am Beispiel existierender Zeilen eines Vergleichsbegriffs (z.B. Konto-Nr. 100) zweier Tabellen. Es wird bei den Outer Joins unterstellt, dass die Table 1 in der LEFT-Position ist (der dargestellte Wert entspricht der Anzahl existierender Zeilen):

Datenbasis		Ergebniszeilen		
Table 1 (Left)	Table 2 (Right)	Inner Join	Left Outer	Right Outer
0	0	0	0	0
1	0	0	1	0
0	1	0	0	1
1	1	1	1	1
2	0	0	2	0
2	1	2	2	2
2	2	4	4	4
n	**0**	**0**	**n**	**0**
0	**n**	**0**	**0**	**n**
n (>0)	**m (>0)**	**n*m**	**n*m**	**n*m**

Wenn mehrere Joins stattfinden, repräsentiert die Table 1 die bisherigen zusammengemischten Tabellen.

Ein **INNER JOIN** fasst nur die Zeilen zusammen, die übereinstimmende Werte der Vergleichsbegriffe aufweisen. Dies sind i.d.R. PK- und FK-Werte.
Filterungen sollten grundsätzlich in der WHERE-Bedingung aufgeführt werden.
Ausnahmen können sich ggf. beim Einsatz von Nested Table Epressions ergeben, da aufgrund der Syntax-Anforderungen dort die Reihenfolge der definierten Tabellen eine Rolle spielen kann.
Dort können dann ggf. erforderliche Filterungen auch ausnahmsweise mit der ON-Klausel vorgegeben werden, ansonsten muss in der FROM-Klausel der Einsatz des Schlüsselworts TABLE eruiert werden.

Ein **OUTER JOIN** fasst wie der INNER JOIN die Zeilen zusammen, die übereinstimmende Werte der Vergleichsbegriffe aufweisen. Dies sind i.d.R. PK- und FK-Werte. Zusätzlich wird auch für jede nicht übereinstimmende Zeile eine Wertegruppe mit ausschließlich NULL-Werten* gebildet und zu der Ergebnismenge hinzugefügt.

Bei einem Outer Join gelten folgende Regeln:

- Die Filterung, d.h. Auswahl der Daten findet mit der **WHERE**-Klausel statt.
  Sind WHERE-Bedingungen vorgegeben, erfüllen alle Zeilen der Result Table diese Bedingungen (WHERE definiert Filter-Bedingungen).

- Die **ON-Klausel** spezifiziert die Übereinstimmungs-Bedingungen. Nicht übereinstimmende Zeilen werden komplett als NULL-Werte* erzeugt.
  Die ON-Klausel spezifiziert keine Filter-Bedingungen.

Vor der Outer-Join-Bedingung wirkt zunächst immer ein Inner Join. Anschließend werden bei Nicht-Übereinstimmung die entsprechenden NULL-Werte erzeugt.

** NULL-Werte können mit der COLAESCE-Funktion verändert werden.*

Ein Join führt die Spalten zweier Tabellen logisch auf horizontaler Ebene zusammen. Somit verschmelzen zwei Strukturen bzw. Struktur-Ausschnitte (Projektion). Dies ist das Gegenteil eines UNIONs oder anderen SET-Operators, bei dem die Strukturen bzw. Struktur-Ausschnitte (Projektion) identisch oder zumindest kompatibel sind. Hier findet eine Zusammenführung von Zeilen in vertikaler Ebene statt.

Die Dateninhalte (Zeilen-Werte) werden i.d.R. mit Equi-Join-Technik zusammengeführt, das heißt die Zeilen der einen Tabelle (PK) werden mit den korrelierenden Zeilen der anderen Tabelle (FK) zusammengemischt.
Die grundsätzlichen Effekte eines Joins entsprechen dabei einer Multiplikation mit der Wirkung auf Zeilen-Ebene:

- Ein **Inner-Join** multipliziert die übereinstimmenden Zeilen (PK <--> FK), wobei ein nicht übereinstimmender Wert (nicht existierend) in einer Tabelle mit 0 Zeilen (eigentlich NULL) angerechnet wird:

  1 * 0 = 0 1 * 1 = 1 1 * 2 = 2 usw.

  Bei einem Inner Join werden auch bis zu einem bestimmten Zeitpunkt selektierte Daten wieder ausgefiltert, wenn eine der beiden Vergleichszeilen keinen übereinstimmenden Wert aufweist!

- Ein **Outer-Join** multipliziert die übereinstimmenden Zeilen (PK <--> FK), wobei ein nicht übereinstimmender Wert (nicht existierend) in einer Tabelle mit pro gegenüberstehender Zeile mit 1 Zeile angerechnet wird, die ausschließlich aus NULL-Werten besteht (sofern diese nicht mit COALESCE behandelt werden):

  1 * 0 (=1) = 1 1 * 1 = 1 1 * 2 = 2 usw.

  Ein Outer Join erhält die bis zu einem bestimmten Zeitpunkt selektierten Daten auch dann, wenn eine der beiden Vergleichszeilen keinen übereinstimmenden Wert aufweist!

Bei all diesen Zusammenführungen spielen wiederum die Daten-Modell-Varianten eine bestimmte Rolle. So wird i.d.R. in einer typischen Hierarchie nur einmal ein PK auftreten und ggf. mehrfache Zeilen mit übereinstimmendem FK.

Zusammenfassung der Effekte hinsichtlich der Zeilen-Anzahl (L =Left Table; R = Right Table):

- **INNER JOIN** **multipliziert vorhandene übereinstimmende Zeilen (3L * 4R = 12), aber eliminiert wiederum Zeilen, wenn auf der einen Seite keine identischen Werte auftreten (3L * 4R = 12L * 0R = 0).**
  So kann ein Inner Join zu sehr hohen Zeilen-Anzahlen in Zwischenschritten führen, die dann aber später wieder eliminiert werden können.

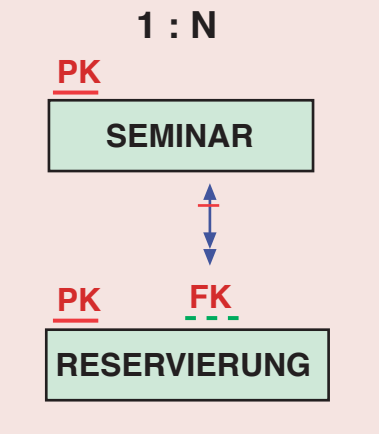
- **LEFT JOIN** **multipliziert zunächst vorhandene übereinstimmende Zeilen (3L * 4R = 12) und erzeugt für die rechte Tabelle NULL-Wert-Zeilen, wenn dort keine identischen Werte auftreten ((3L * 4R) = 12L + (2L * 1 R) = 14).**
  Damit kann speziell der Left Outer Join einmal gebildete Zeilen-Anzahlen erhalten, die dann auch später durch nicht existierende Right-Table-Zeilen nicht mehr eliminiert werden.
  Dies gilt insbesondere für Left Outer Joins, da die Ergebnis-Interims-Tabelle eines Join-Schrittes immer in der Left-Rolle zur neuen Tabelle eines nächsten Join-Schritts auftritt.

- **RIGHT JOIN** **multipliziert zunächst vorhandene übereinstimmende Zeilen (3L * 4R = 12) und erzeugt für die linke Tabelle NULL-Wert-Zeilen, dort keine identischen Werte auftreten ((3L * 4R) = 12L + (1L * 3R) = 15).**
  Dieser Join-Typ eliminiert aber wiederum Zeilen, wenn auf der rechten Seite keine identischen Werte auftreten ((3L * 4R) = 12L * 0R = 0).
  Die übereinstimmenden Zeilen bleiben natürlich erhalten.

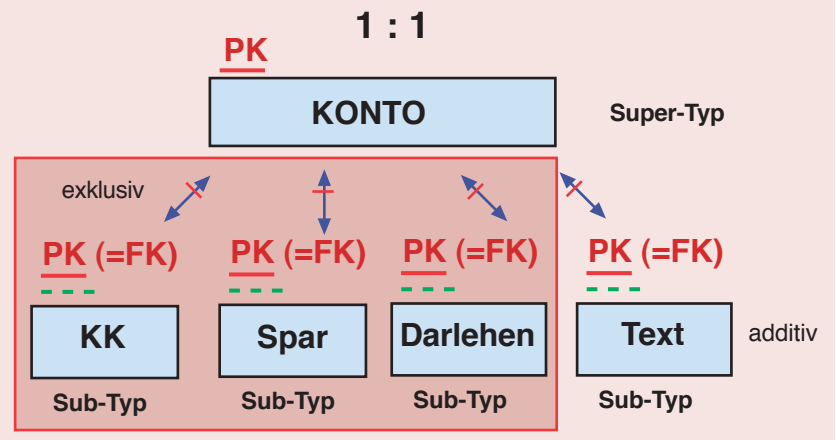
1.1.3 Hierarchie-Typen

1.1.3.1 Subset-Hierarchie und Generalisierte Hierarchie

Subset-Hierarchie:



Generalisierte Hierarchie



Regeln:

- Der Foreign-Key kann Teil des PKs sein. In jedem Fall ist er nicht NULL-fähig.
- Der Foreign-Key kann grundsätzlich Duplikate aufweisen (Not-Unique); Ausnahme: bei der generalisierten Hierarchie.

Als typische hierarchische Beziehung kann die **Subset-Hierarchie** bezeichnet werden, die sich über mehrere Stufen erstrecken kann.
Hier existiert nur ein Wert pro PK (dieser ist ja auch Unique) und 0, 1 oder n korellierende Zeilen mit entsprechend identischem FK-Wert.

Ausnahmen von dieser Regel existieren z.B. bei **Modellen mit Versionierung oder Zeitkonzepten**, bei denen keine Identität der Werte existiert; in solchen Fällen müssen sich z.B. die Daten einem Zeitraum zuordnen lassen (von - bis).

Für das Zusammenführen der Informationen steht die SQL-Variante zur Verfügung:

- **Alle JOIN-Varianten. Inner und Outer Join, bezogen auf die abhängige Tabelle erbringen keine unterschiedlichen Ergebnisse.**

Eine besondere, allerdings in der Praxis sehr häufig anzutreffende Form ist die Generalisierte Hierarchie, bei der exklusive Typen definierbar sind mit folgenden Besonderheiten:

- **Jedes Objekt** verfügt über **denselben PK-Key-Aufbau**

- **Ein Super-Typ** definiert die **gemeinsamen Charakteristiken** und ist im wesentlichen auch verantwortlich für die Vergabe des eindeutigen PKs aller Objekte in der Hierarchie.

- Es können ein bis n **Sub-Typen** mit **individuellen Charakteristiken** definiert werden.
  Innerhalb der Sub-Typen sind die PKs exklusiv, d.h. ein bestimmter PK-Wert darf nur in einem Sub-Typen auftreten.

Das nebenstehende Beispiel zeigt einen einfachen Auszug eines Daten-Modells aus dem Bankenbereich mit dem Super-Typ Konto und den Sub-Typen Kontokorrent-Konto, Spar-Konto und Darlehens-Konto.

Die Generalisierte Hierarchie kann mit Subset-Hierarchie-Komponenten gemischt werden. Dies ist auch in unserer nebenstehenden Abbildung der Fall. Der Text kann additiv bei allen Typen auftreten.
In unserem Beispiel könnte der Text auch dem Super-Typ zugeordnet werden. Eine Auslagerung in einen eigenen Sub-Typ fördert die generische Nutzungs-Möglichkeit solcher Text-Objekte.

Die Generalisierte Hierarchie lässt sich im Relationen-Modell nicht explizit abbilden. Die besonderen Regeln müssen applikatorisch überwacht werden.
Für das Zusammenführen der Informationen stehen folgende SQL-Varianten zur Verfügung:

- **UNION, EXCEPT, INTERSECT** (alle Varianten mit oder ohne **ALL**)
- **JOIN-Varianten: speziell der Outer Join; Inner Join nur bei Teil-Daten-Filterung (auf Sub-Typ-Ebene).**

1.1.3.2 Subset-Hierarchie

1.1.3.2.1 Inner Join

Parent P

```
  select * from session.parent ;
  +---------------------------------+
  |      PID    | P_TYP |   P_TEXT    |
  +---------------------------------+
1_|          1  | A     | Parent 1    |
2_|          2  | B     | Parent 2    |
3_|          4  | B     | Parent 4    |
4_|          5  | A     | Parent 5    |
  +---------------------------------+
0SUCCESSFUL RETRIEVAL OF     4 ROW(S)
```

Child C

```
  select * from session.child ;
  +-------------------------------------------+
  |    CID     |    PID     | C_TYP |   C_TEXT    |
  +-------------------------------------------+
1_|        11  |          1 | X     | Child 1 1   |
2_|        12  |          1 | Y     | Child 1 2   |
3_|        41  |          4 | X     | Child 4 1   |
4_|        51  |          5 | X     | Child 5 1   |
  +-------------------------------------------+
0SUCCESSFUL RETRIEVAL OF     4 ROW(S)
```

Bei einem INNER JOIN werden grundsätzlich die Parent Rows mit ihrem PK zu den Dependent Rows mit den FK-Werten zugeordnet.
Es können aber auch komplexe Suchbedingungen inkl. Sub-Select vorgegeben werden.

Filterungen sollten grundsätzlich in der WHERE-Bedingung aufgeführt werden.

```
-- Inner Join komplett
  select  p.* , c.*
    from         session.parent as p
      inner join session.child  as c
        on  p.pid = c.pid
    ;
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
        PID  P_TYP  P_TEXT              CID          PID  C_TYP  C_TEXT
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
          1  A      Parent 1             11            1  X      Child 1 1
          1  A      Parent 1             12            1  Y      Child 1 2
          4  B      Parent 4             41            4  X      Child 4 1
          5  A      Parent 5             51            5  X      Child 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 4
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
```

```
-- Inner Join mit Filterung
   select  p.* , c.*
    from         session.parent as p
      inner join session.child  as c
        on  p.pid = c.pid
        and p.p_typ = 'A'
    ;
----+----+----+----+----+----+----+----+----+----+----+----+----+----+--------
        PID  P_TYP  P_TEXT              CID          PID  C_TYP  C_TEXT
----+----+----+----+----+----+----+----+----+----+----+----+----+----+--------
          1  A      Parent 1             11            1  X      Child 1 1
          1  A      Parent 1             12            1  Y      Child 1 2
          5  A      Parent 5             51            5  X      Child 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 3
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
----+----+----+----+----+----+----+----+----+----+----+----+----+----+--------
   select  p.* , c.*
    from         session.parent as p
      inner join session.child  as c
        on  p.pid = c.pid
      where p.p_typ = 'A'
    ;
----+----+----+----+----+----+----+----+----+----+----+----+----+----+--------
        PID  P_TYP  P_TEXT              CID          PID  C_TYP  C_TEXT
----+----+----+----+----+----+----+----+----+----+----+----+----+----+--------
          1  A      Parent 1             11            1  X      Child 1 1
          1  A      Parent 1             12            1  Y      Child 1 2
          5  A      Parent 5             51            5  X      Child 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 3
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
----+----+----+----+----+----+----+----+----+----+----+----+----+----+--------
   select  p.* , c.*
    from         session.parent as p
      inner join session.child  as c
        on  p.pid = c.pid
        and c.c_typ = 'X'
    ;
----+----+----+----+----+----+----+----+----+----+----+----+----+----+--------
        PID  P_TYP  P_TEXT              CID          PID  C_TYP  C_TEXT
----+----+----+----+----+----+----+----+----+----+----+----+----+----+--------
          1  A      Parent 1             11            1  X      Child 1 1
          4  B      Parent 4             41            4  X      Child 4 1
          5  A      Parent 5             51            5  X      Child 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 3
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
----+----+----+----+----+----+----+----+----+----+----+----+----+----+--------
   select  p.* , c.*
    from         session.parent as p
      inner join session.child  as c
        on  p.pid = c.pid
      where c.c_typ = 'X'
    ;
----+----+----+----+----+----+----+----+----+----+----+----+----+----+--------
        PID  P_TYP  P_TEXT              CID          PID  C_TYP  C_TEXT
----+----+----+----+----+----+----+----+----+----+----+----+----+----+--------
          1  A      Parent 1             11            1  X      Child 1 1
          4  B      Parent 4             41            4  X      Child 4 1
          5  A      Parent 5             51            5  X      Child 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 3
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
----+----+----+----+----+----+----+----+----+----+----+----+----+----+--------
```

Fazit: Beim Inner Join wirken ON-Klausel und WHERE äquivalant auf die Datenauswahl.

1.1.3.2.2 Left Outer Join

Parent P

```
  select * from session.parent ;
  +-------------------------------+
  |      PID    | P_TYP |   P_TEXT    |
  +-------------------------------+
1_|           1 | A     | Parent 1    |
2_|           2 | B     | Parent 2    |
3_|           4 | B     | Parent 4    |
4_|           5 | A     | Parent 5    |
  +-------------------------------+
0SUCCESSFUL RETRIEVAL OF     4 ROW(S)
```

Child C

```
  select * from session.child ;
  +-------------------------------------------+
  |    CID     |     PID     | C_TYP |   C_TEXT    |
  +-------------------------------------------+
1_|         11 |           1 | X     | Child 1 1   |
2_|         12 |           1 | Y     | Child 1 2   |
3_|         41 |           4 | X     | Child 4 1   |
4_|         51 |           5 | X     | Child 5 1   |
  +-------------------------------------------+
0SUCCESSFUL RETRIEVAL OF     4 ROW(S)
```

Der Left Outer Join führt zunächst logisch einen Inner Join durch. Dann werden alle nicht-übereinstimmenden Zeilen (hier: PID 2 der Parent-Tabelle) ebenfalls in die Result Table einbezogen.
Da für diesen PK keine Zeilen mit identischem FK existieren, werden die Spalten-Werte der Dependant Table mit NULL-Werten aufgefüllt. Dies bezieht sich auch auf Spalten, die per Definition nicht NULL-fähig sind, wie z.B. der PK CID.

Wird die ON-Klausel (neben dem PK-FK) erweitert, erweitert sich die Bedingung zur Erkennung der Gleichheit zweier Zeilen entsprechend. Siehe hierzu das erste Beispiel auf der rechten Seite.
Für Left Outer Join und Right Outer Join können in der ON-Klausel komplexe Suchbedingungen inkl. Sub-Select vorgegeben werden.

Wie vorab ausgeführt, entscheidet nur die WHERE-Klausel, nicht aber die ON-Klausel über die bereitgestellte Daten-Menge.

```
-- left outer Join komplett
  select  p.* , c.*
    from         session.parent as p
      left  join session.child  as c
        on  p.pid = c.pid
    ;
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
        PID  P_TYP  P_TEXT              CID          PID  C_TYP  C_TEXT
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
          1  A      Parent 1             11            1  X      Child 1 1
          1  A      Parent 1             12            1  Y      Child 1 2
          2  B      Parent 2    -----------  -----------  -----  ----------
          4  B      Parent 4             41            4  X      Child 4 1
          5  A      Parent 5             51            5  X      Child 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 5
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
```

```
-- left  Join mit Filterung
   select  p.* , c.*
    from         session.parent as p
      left  join session.child  as c
        on  p.pid = c.pid
        and p.p_typ = 'A'
    ;
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
        PID  P_TYP  P_TEXT              CID          PID  C_TYP  C_TEXT
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
          1  A      Parent 1             11           1  X      Child 1 1
          1  A      Parent 1             12           1  Y      Child 1 2
          2  B      Parent 2    -----------  -----------  -----  ----------
          4  B      Parent 4    -----------  -----------  -----  ----------
          5  A      Parent 5             51           5  X      Child 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 5
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
   select  p.* , c.*
    from         session.parent as p
      left  join session.child  as c
        on  p.pid = c.pid
      where p.p_typ = 'A'
    ;
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
        PID  P_TYP  P_TEXT              CID          PID  C_TYP  C_TEXT
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
          1  A      Parent 1             11           1  X      Child 1 1
          1  A      Parent 1             12           1  Y      Child 1 2
          5  A      Parent 5             51           5  X      Child 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 3
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
   select  p.* , c.*
    from         session.parent as p
      left  join session.child  as c
        on  p.pid = c.pid
        and c.c_typ = 'X'
    ;
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
        PID  P_TYP  P_TEXT              CID          PID  C_TYP  C_TEXT
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
          1  A      Parent 1             11           1  X      Child 1 1
          2  B      Parent 2    -----------  -----------  -----  ----------
          4  B      Parent 4             41           4  X      Child 4 1
          5  A      Parent 5             51           5  X      Child 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 4
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
   select  p.* , c.*
    from         session.parent as p
      left  join session.child  as c
        on  p.pid = c.pid
      where c.c_typ = 'X'
    ;
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
        PID  P_TYP  P_TEXT              CID          PID  C_TYP  C_TEXT
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
          1  A      Parent 1             11           1  X      Child 1 1
          4  B      Parent 4             41           4  X      Child 4 1
          5  A      Parent 5             51           5  X      Child 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 3
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
```

Fazit: Beim Outer Join wirken ON-Klausel und WHERE unterschiedlich auf die Datenauswahl.

1.1.3.2.3 Right Outer Join

Parent P

```
  select * from session.parent ;
  +---------------------------------+
  |      PID    | P_TYP |   P_TEXT    |
  +---------------------------------+
1_|          1  | A     | Parent 1    |
2_|          2  | B     | Parent 2    |
3_|          4  | B     | Parent 4    |
4_|          5  | A     | Parent 5    |
  +---------------------------------+
0SUCCESSFUL RETRIEVAL OF     4 ROW(S)
```

Child C

```
  select * from session.child ;
  +-------------------------------------------+
  |    CID     |     PID     | C_TYP |   C_TEXT    |
  +-------------------------------------------+
1_|        11  |          1  | X     | Child 1 1   |
2_|        12  |          1  | Y     | Child 1 2   |
3_|        41  |          4  | X     | Child 4 1   |
4_|        51  |          5  | X     | Child 5 1   |
  +-------------------------------------------+
0SUCCESSFUL RETRIEVAL OF     4 ROW(S)
```

Der Right Outer Join entspricht in unserem Beispiel zunächst grundsätzlich einem Inner Join, sofern neben dem PK-FK keine weiteren Filterungen bzw. Erweiterungen in der ON-Klausel auftreten.

Für Left Outer Join und Right Outer Join können in der ON-Klausel komplexe Suchbedingungen inkl. Sub-Select vorgegeben werden.

```
-- right outer Join komplett
  select  p.* , c.*
    from         session.parent as p
      right join session.child  as c
        on  p.pid = c.pid
    ;
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
        PID  P_TYP  P_TEXT              CID          PID  C_TYP  C_TEXT
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
          1  A      Parent 1             11            1  X      Child 1 1
          1  A      Parent 1             12            1  Y      Child 1 2
          4  B      Parent 4             41            4  X      Child 4 1
          5  A      Parent 5             51            5  X      Child 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 4
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
```

```
-- right Join mit Filterung
   select  p.* , c.*
    from         session.parent as p
      right join session.child  as c
        on  p.pid = c.pid
        and p.p_typ = 'A'
    ;
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
        PID  P_TYP  P_TEXT              CID          PID  C_TYP  C_TEXT
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
          1  A      Parent 1             11           1  X      Child 1 1
          1  A      Parent 1             12           1  Y      Child 1 2
-----------  -----  ----------           41           4  X      Child 4 1
          5  A      Parent 5             51           5  X      Child 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 4
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
   select  p.* , c.*
    from         session.parent as p
      right join session.child  as c
        on  p.pid = c.pid
      where p.p_typ = 'A'
    ;
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
        PID  P_TYP  P_TEXT              CID          PID  C_TYP  C_TEXT
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
          1  A      Parent 1             11           1  X      Child 1 1
          1  A      Parent 1             12           1  Y      Child 1 2
          5  A      Parent 5             51           5  X      Child 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 3
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
   select  p.* , c.*
    from         session.parent as p
      right join session.child  as c
        on  p.pid = c.pid
        and c.c_typ = 'X'
    ;
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
        PID  P_TYP  P_TEXT              CID          PID  C_TYP  C_TEXT
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
          1  A      Parent 1             11           1  X      Child 1 1
-----------  -----  ----------           12           1  Y      Child 1 2
          4  B      Parent 4             41           4  X      Child 4 1
          5  A      Parent 5             51           5  X      Child 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 4
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
   select  p.* , c.*
    from         session.parent as p
      right join session.child  as c
        on  p.pid = c.pid
      where c.c_typ = 'X'
    ;
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
        PID  P_TYP  P_TEXT              CID          PID  C_TYP  C_TEXT
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
          1  A      Parent 1             11           1  X      Child 1 1
          4  B      Parent 4             41           4  X      Child 4 1
          5  A      Parent 5             51           5  X      Child 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 3
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
```

Fazit: Beim Outer Join wirken ON-Klausel und WHERE unterschiedlich auf die Datenauswahl.

1.1.3.2.4 Full Outer Join

Parent P

```
  select * from session.parent ;
  +-------------------------------+
  |      PID   | P_TYP |   P_TEXT   |
  +-------------------------------+
1_|          1 | A     | Parent 1   |
2_|          2 | B     | Parent 2   |
3_|          4 | B     | Parent 4   |
4_|          5 | A     | Parent 5   |
  +-------------------------------+
0SUCCESSFUL RETRIEVAL OF     4 ROW(S)
```

Child C

```
  select * from session.child ;
  +-----------------------------------------+
  |    CID    |    PID    | C_TYP |  C_TEXT   |
  +-----------------------------------------+
1_|        11 |         1 | X     | Child 1 1 |
2_|        12 |         1 | Y     | Child 1 2 |
3_|        41 |         4 | X     | Child 4 1 |
4_|        51 |         5 | X     | Child 5 1 |
  +-----------------------------------------+
0SUCCESSFUL RETRIEVAL OF     4 ROW(S)
```

Der FULL Outer Join entspricht in unserem Beispiel einem LEFT Outer Join.
Bei einem FULL Outer Join sind nicht alle ON-Spezfikationen unterstützt. So sind nur Equal-Operatoren und Referenzen auf eine Join-Tabelle, nicht aber Konstanten, Host-Variablen-Bezüge oder die Vorgabe komplexer Suchbedingungen möglich.

```
 -- full outer Join komplett
  select  p.* , c.*
    from         session.parent as p
      full join session.child   as c
        on  p.pid = c.pid
    ;
-------------------------------------------------------------------------------
        PID  P_TYP  P_TEXT              CID          PID  C_TYP  C_TEXT
-------------------------------------------------------------------------------
          1  A      Parent 1             12            1  Y      Child 1 2
          1  A      Parent 1             11            1  X      Child 1 1
          2  B      Parent 2    -----------  -----------  -----  ----------
          4  B      Parent 4             41            4  X      Child 4 1
          5  A      Parent 5             51            5  X      Child 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 5
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
```

```
-- full Join mit Filterung
-- select  p.* , c.*
--  from         session.parent as p
--    full join session.child   as c
--      on  p.pid = c.pid
--      and p.p_typ = 'A'    /* unzulaessig im Full Join */
--  ;
   select  p.* , c.*
    from         session.parent as p
      full join session.child   as c
        on  p.pid = c.pid
      where p.p_typ = 'A'
    ;
------------------------------------------------------------------------------
        PID  P_TYP  P_TEXT              CID          PID  C_TYP  C_TEXT
------------------------------------------------------------------------------
          1  A      Parent 1             11            1  X      Child 1 1
          1  A      Parent 1             12            1  Y      Child 1 2
          5  A      Parent 5             51            5  X      Child 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 3
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
------------------------------------------------------------------------------
-- select  p.* , c.*
--  from         session.parent as p
--    full join session.child   as c
--      on  p.pid = c.pid
--      and c.c_typ = 'X'      /* unzulaessig im Full Join */
--  ;
   select  p.* , c.*
    from         session.parent as p
      full join session.child   as c
        on  p.pid = c.pid
      where c.c_typ = 'X'
    ;
------------------------------------------------------------------------------
        PID  P_TYP  P_TEXT              CID          PID  C_TYP  C_TEXT
------------------------------------------------------------------------------
          1  A      Parent 1             11            1  X      Child 1 1
          4  B      Parent 4             41            4  X      Child 4 1
          5  A      Parent 5             51            5  X      Child 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 3
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
------------------------------------------------------------------------------
```

1.1.3.3 Generalisierte Hierarchie

1.1.3.3.1 Inner Join

Super-Typ ST

```
  select * from session.supertyp ;
  +-----------------------------+
  |      PID       |    TEXT    |
  +-----------------------------+
1_|              1 | supertyp 1 |
2_|              2 | supertyp 2 |
3_|              3 | supertyp 3 |
4_|              4 | supertyp 4 |
5_|              5 | supertyp 5 |
  +-----------------------------+
0SUCCESSFUL RETRIEVAL OF  5 ROW(S)
```

Sub-Typen

```
  select * from session.subtyp_kk ;
  +-------------------------+
  |      PID       | KK_TYP |
  +-------------------------+
1_|              1 | G      |
2_|              4 | T      |
  +-------------------------+
SUCCESSFUL RETRIEVAL OF 2 ROW(S)

select * from session.subtyp_sp ;
  +-----------------------------+
  |      PID       | SP_VARIANTE |
  +-----------------------------+
1_|              2 | A           |
2_|              5 | B           |
+-----------------------------+
0SUCCESSFUL RETRIEVAL OF  2 ROW(S)

  select * from session.subtyp_dl ;
  +-------------------------+
  |      PID       | DL_TYP |
  +-------------------------+
1_|              3 | H      |
  +-------------------------+
SUCCESSFUL RETRIEVAL OF  1 ROW(S)

    select * from session.subtyp_tx ;
  +-------------------------------------------------+
  |      PID       |              TEXT              |
  +-------------------------------------------------+
1_|              1 | Kontokorrentkonto 1            |
2_|              2 | Sparkonto         2            |
3_|              3 | Darlehenskonto    3            |
4_|              4 | Kontokorrentkonto 4            |
  +-------------------------------------------------+
SUCCESSFUL RETRIEVAL OF 4 ROW(S)
```

Ein INNER JOIN macht nur unter bestimmten Bedingungen Sinn, wie:

- Die Daten werden nur aus dem Blickwinkel eines Sub-Typs betrachtet.
- Die Daten verschiedener Result Tables werden unter Zuhilfenahmen eines Set-Operators (wie UNION) zusammengeführt.

```
-- Inner Join komplett

    select  st.* , kk.* , sp.* , dl.* , tx.*
      from         session.supertyp as st
        inner join session.subtyp_kk as kk
          on  st.pid = kk.pid
        inner join session.subtyp_sp as sp
          on  st.pid = sp.pid
        inner join session.subtyp_dl as dl
          on  st.pid = dl.pid
        inner join session.subtyp_tx as tx
          on  st.pid = tx.pid
     ;
 SUCCESSFUL RETRIEVAL OF        0 ROW(S)
```

Fazit: Beim Inner Join können nicht alle Sub-Typen eingebunden werden.

```
   -- Inner Join komplett mit UNION
    select  st.* , kk.*
      from         session.supertyp as st
        inner join session.subtyp_kk as kk
          on  st.pid = kk.pid
```

```
    union
    select  st.* ,  sp.*
      from         session.supertyp as st
        inner join session.subtyp_sp as sp
          on  st.pid = sp.pid
    union
    select  st.* ,  st.*
      from         session.supertyp as st
        inner join session.subtyp_dl as dl
          on  st.pid = dl.pid
    union
    select  st.* ,  tx.*
      from         session.supertyp as st
        inner join session.subtyp_tx as tx
          on  st.pid = tx.pid
      ;
   +-------------------------------------------------------------------------------+
   |      PID       |    TEXT    |      PID       |                                |
   +-------------------------------------------------------------------------------+
1_|              1 | supertyp 1 |              1 | G                              |
2_|              1 | supertyp 1 |              1 | Kontokorrentkonto 1            |
3_|              2 | supertyp 2 |              2 | A                              |
4_|              2 | supertyp 2 |              2 | Sparkonto         2            |
5_|              3 | supertyp 3 |              3 | Darlehenskonto    3            |
6_|              3 | supertyp 3 |              3 | supertyp 3                     |
7_|              4 | supertyp 4 |              4 | Kontokorrentkonto 4            |
8_|              4 | supertyp 4 |              4 | T                              |
9_|              5 | supertyp 5 |              5 | B                              |
   +-------------------------------------------------------------------------------+
 SUCCESSFUL RETRIEVAL OF  9 ROW(S)
```

Fazit: Ein UNION oder besser UNION ALL führt die Daten, soweit kompatibel zeilenweise zusammen.

1.1.3.3.2 Left Outer Join

Super-Typ ST

```
select * from session.supertyp ;
+-----------------------------+
|      PID       |    TEXT    |
+-----------------------------+
1_|              1 | supertyp 1 |
2_|              2 | supertyp 2 |
3_|              3 | supertyp 3 |
4_|              4 | supertyp 4 |
5_|              5 | supertyp 5 |
  +-----------------------------+
0SUCCESSFUL RETRIEVAL OF  5 ROW(S)
```

Sub-Typen

```
select * from session.subtyp_kk ;
  +-------------------------+
  |      PID       | KK_TYP |
  +-------------------------+
1_|              1 | G      |
2_|              4 | T      |
  +-------------------------+
SUCCESSFUL RETRIEVAL OF 2 ROW(S)

select * from session.subtyp_sp ;
  +-------------------------------+
  |      PID       | SP_VARIANTE  |
  +-------------------------------+
1_|              2 | A            |
2_|              5 | B            |
+-------------------------------+
0SUCCESSFUL RETRIEVAL OF  2 ROW(S)

  select * from session.subtyp_dl ;
  +-------------------------+
  |      PID       | DL_TYP |
  +-------------------------+
1_|              3 | H      |
  +-------------------------+
SUCCESSFUL RETRIEVAL OF  1 ROW(S)

    select * from session.subtyp_tx ;
  +-------------------------------------------------+
  |      PID       |              TEXT              |
  +-------------------------------------------------+
1_|              1 | Kontokorrentkonto 1            |
2_|              2 | Sparkonto         2            |
3_|              3 | Darlehenskonto    3            |
4_|              4 | Kontokorrentkonto 4            |
  +-------------------------------------------------+
SUCCESSFUL RETRIEVAL OF 4 ROW(S)
```

Der Left Outer Join führt zunächst logisch einen Inner Join durch. Dies betrifft die Zeilen mit PID 1 und 4. Dann werden alle nicht-übereinstimmenden Zeilen (hier: PID 2, 3 und 5 der Super-Typ-Tabelle) ebenfalls in die Result Table einbezogen.
Da für diese letzteren PKs keine Zeilen mit identischem FK in der Subtyp_KK-Tabelle existieren, werden die Spalten-Werte der Dependent Table mit NULL-Werten aufgefüllt.
Zu diesem Zeitpunkt besteht die virtuelle Interimstabelle aus 4 Spalten (und 5 Zeilen).
Diese wird dann aufgrund des nächsten Left Outer Joins um weitere 2 Spalten erweitert, nämlich den Spalten aus der Subtyp_SP-Tabelle. Nun werden alle Werte übereinstimmenden Zeilen dort eingestellt, die der nicht übereinstimmenden mit NULL-Werten versehen.

Und so entwickelt sich Schritt-für-Schritt die Interimstabelle durch ständige horizontale Struktur-Erweiterungen zur endgültigen Result Table.
Aufgrund der Left-Outer-Logik wird hier das Gesamt-Informationsspektrum - repräsentiert durch den Super-Typ - einmal aufgebaut und dann Zug-um-Zug ergänzt. Es werden dadurch später auch keine Zeilen mehr eliminiert, was bei einem Inner Join der Fall wäre.
Die 5 Zeilen der Ergebnis-Tabelle werden bereits im ersten Schritt gebildet.

```
-- Left Outer Join komplett
 ***INPUT STATEMENT:
    select  st.* , kk.* , sp.* , dl.* , tx.*
      from         session.supertyp as st
        left  join session.subtyp_kk as kk
          on  st.pid = kk.pid
        left  join session.subtyp_sp as sp
          on  st.pid = sp.pid
        left  join session.subtyp_dl as dl
          on  st.pid = dl.pid
        left  join session.subtyp_tx as tx
          on  st.pid = tx.pid
      ;
  +-------------------------------------------------------------------------------------------
  |   PID   |    TEXT    |   PID   | KK_TYP |   PID     | SP_VARIANTE |   PID     | DL_TYP |
  +-------------------------------------------------------------------------------------------
1_|      1 | supertyp 1 |       1 | G      | ?         | ?           | ?         | ?      |
2_|      2 | supertyp 2 | ?       | ?      |         2 | A           | ?         | ?      |
3_|      3 | supertyp 3 | ?       | ?      | ?         | ?           |         3 | H      |
4_|      4 | supertyp 4 |       4 | T      | ?         | ?           | ?         | ?      |
5_|      5 | supertyp 5 | ?       | ?      |         5 | B           | ?         | ?      |
+-------------------------------------------------------------------------------------------
1PAGE    2
               ------------------------------------------------+
               |      PID       |              TEXT             |
               ------------------------------------------------+
            1_|              1 | Kontokorrentkonto 1           |
            2_|              2 | Sparkonto         2           |
            3_|              3 | Darlehenskonto    3           |
            4_|              4 | Kontokorrentkonto 4           |
            5_| ?              | ?                             |
              ------------------------------------------------+
 SUCCESSFUL RETRIEVAL OF        5 ROW(S)
```

1.1.3.3.3 Variante: Redundantes Sub-Typ-Merkmal in Super-Typ

Super-Typ ST

```
select * from session.supertyp ;
 +------------------------------
 |  PID  | TYP |    TEXT     |
 +------------------------------
1_|     1 | KK  | supertyp 1 |
2_|     2 | SP  | supertyp 2 |
3_|     3 | DL  | supertyp 3 |
4_|     4 | KK  | supertyp 4 |
5_|     5 | SP  | supertyp 5 |
 +------------------------------
0SUCCESSFUL RETRIEVAL OF  5 ROW(S)
```

Sub-Typen

```
select * from session.subtyp_kk ;
 +-------------------------+
 |      PID       | KK_TYP |
 +-------------------------+
1_|             1 | G      |
2_|             4 | T      |
 +-------------------------+
SUCCESSFUL RETRIEVAL OF 2 ROW(S)

select * from session.subtyp_sp ;
 +------------------------------+
 |      PID       | SP_VARIANTE |
 +------------------------------+
1_|             2 | A           |
2_|             5 | B           |
+------------------------------+
0SUCCESSFUL RETRIEVAL OF  2 ROW(S)

select * from session.subtyp_dl ;
 +-------------------------+
 |      PID       | DL_TYP |
 +-------------------------+
1_|             3 | H      |
 +-------------------------+
SUCCESSFUL RETRIEVAL OF  1 ROW(S)

   select * from session.subtyp_tx ;
 +-------------------------------------------------+
 |      PID       |              TEXT              |
 +-------------------------------------------------+
1_|             1 | Kontokorrentkonto 1            |
2_|             2 | Sparkonto         2            |
3_|             3 | Darlehenskonto    3            |
4_|             4 | Kontokorrentkonto 4            |
 +-------------------------------------------------+
SUCCESSFUL RETRIEVAL OF 4 ROW(S)
```

Diese Erweiterung, die häufig in der Praxis anzutreffen ist, zeigt keine Veränderungen in der SQL-Nutzungs-Möglichkeit:

```
  -- Inner Join komplett mit erweiterter ON-Klausel
 ***INPUT STATEMENT:
    select  st.* , kk.* , sp.* , dl.* , tx.*
      from         session.supertyp as st
        inner join session.subtyp_kk as kk
          on  st.pid = kk.pid
          and st.typ = 'KK'
        inner join session.subtyp_sp as sp
          on  st.pid = sp.pid
          and st.typ = 'SP'
        inner join session.subtyp_dl as dl
          on  st.pid = dl.pid
          and st.typ = 'DL'
        inner join session.subtyp_tx as tx
          on  st.pid = tx.pid
      ;
0SUCCESSFUL RETRIEVAL OF        0 ROW(S)
```

```
-- Left Outer Join komplett ohne erweiterte ON-Klausel
    select  st.* , kk.* , sp.* , dl.* , tx.*
      from         session.supertyp as st
        left  join session.subtyp_kk as kk
          on  st.pid = kk.pid
        left  join session.subtyp_sp as sp
          on  st.pid = sp.pid
        left  join session.subtyp_dl as dl
          on  st.pid = dl.pid
        left  join session.subtyp_tx as tx
          on  st.pid = tx.pid
      ;
  +-------------------------------------------------------------------------------------------  |
PID    | TYP |    TEXT     |  PID    | KK_TYP |  PID    | SP_VARIANTE |  PID    | DL_TYP |
  +-------------------------------------------------------------------------------------------1_|
1 | KK   | supertyp 1 |       1 | G       | ?       | ?           | ?       | ?       |
2_|       2 | SP   | supertyp 2 | ?       | ?       |       2 | A           | ?       | ?       |
3_|       3 | DL   | supertyp 3 | ?       | ?       | ?       | ?           |       3 | H       |
4_|       4 | KK   | supertyp 4 |       4 | T       | ?       | ?           | ?       | ?       |
5_|       5 | SP   | supertyp 5 | ?       | ?       |       5 | B           | ?       | ?       |
  +-------------------------------------------------------------------------------------------

                      ---------------------------------------------+
                      |      PID       |              TEXT          |
                      ---------------------------------------------+
                    1_|              1 | Kontokorrentkonto 1        |
                    2_|              2 | Sparkonto         2        |
                    3_|              3 | Darlehenskonto    3        |
                    4_|              4 | Kontokorrentkonto 4        |
                    5_| ?              | ?                          |
                      ---------------------------------------------+
0SUCCESSFUL RETRIEVAL OF        5 ROW(S)
   -- Left Outer Join komplett mit erweiterter ON-Klausel
    select  st.* , kk.* , sp.* , dl.* , tx.*
      from         session.supertyp as st
        left  join session.subtyp_kk as kk
          on  st.pid = kk.pid
          and st.typ = 'KK'
        left  join session.subtyp_sp as sp
          on  st.pid = sp.pid
          and st.typ = 'SP'
        left  join session.subtyp_dl as dl
          on  st.pid = dl.pid
          and st.typ = 'DL'
        left  join session.subtyp_tx as tx
          on  st.pid = tx.pid
      ;
  +-------------------------------------------------------------------------------------------
  |  PID    | TYP |    TEXT     |  PID    | KK_TYP |  PID    | SP_VARIANTE |  PID    | DL_TYP |
  +-------------------------------------------------------------------------------------------
1_|       1 | KK   | supertyp 1 |       1 | G       | ?       | ?           | ?       | ?       |
2_|       2 | SP   | supertyp 2 | ?       | ?       |       2 | A           | ?       | ?       |
3_|       3 | DL   | supertyp 3 | ?       | ?       | ?       | ?           |       3 | H       |
4_|       4 | KK   | supertyp 4 |       4 | T       | ?       | ?           | ?       | ?       |
5_|       5 | SP   | supertyp 5 | ?       | ?       |       5 | B           | ?       | ?       |
  +-------------------------------------------------------------------------------------------

                    ---------------------------------------------+
                    |      PID       |              TEXT          |
                    ---------------------------------------------+
                  1_|              1 | Kontokorrentkonto 1        |
                  2_|              2 | Sparkonto         2        |
                  3_|              3 | Darlehenskonto    3        |
                  4_|              4 | Kontokorrentkonto 4        |
                  5_| ?              | ?                          |
                    ---------------------------------------------+
 SUCCESSFUL RETRIEVAL OF        5 ROW(S)
```

1.1.4 Lockere Beziehung

1.1.4.1 Inner Join

Tabelle 1

```
  select * from session.tab1 ;
------+------+------+------+------+------+
P1ID  P_TYP  P_TEXT
------+------+------+------+------+------+
1     A      tab1 1
2     B      tab1 2
4     B      tab1 4
5     A      tab1 5
DSNE610I NUMBER OF ROWS DISPLAYED IS 4
```

Tabelle 2

```
  select * from session.tab2 ;
------+------+------+------+------+------+-----
T2ID         P1ID  C_TYP  C_TEXT
------+------+------+------+------+------+-----
11            1    X      tab2 1 1
12  ----------     Y      tab2 NL 2
41            4    X      tab2 4 1
51            5    X      tab2 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 4
```

Bei einer lockeren Beziehungen ergeben sich aufgrund der FK-NULL-Regel mehr Daten-Beziehungs-Varianten als bei einer Hierarchie.

Bei einem INNER JOIN werden grundsätzlich die Parent Rows mit ihrem PK zu den Dependent Rows mit den FK-Werten zugeordnet. Es können aber auch Dependent Rows ohne Wert (mit NULL-Wert) auftreten.
Filterungen sollten grundsätzlich in der WHERE-Bedingung aufgeführt werden.

```
-- Inner Join komplett
  select  t1.* , t2.*
    from         session.tab1 as t1
      inner join session.tab2   as t2
        on  t1.P1ID = t2.P1ID
    ;
------+------+------+------+------+------+------+------+------+------+------+---
       P1ID  P_TYP  P_TEXT             T2ID         P1ID  C_TYP  C_TEXT
------+------+------+------+------+------+------+------+------+------+------+---
          1  A      tab1 1               11            1  X      tab2 1 1
          4  B      tab1 4               41            4  X      tab2 4 1
          5  A      tab1 5               51            5  X      tab2 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 3
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
```

```
-- Inner Join mit Filterung
   select  t1.* , t2.*
    from         session.tab1 as t1
      inner join session.tab2 as t2
        on  t1.P1ID = t2.P1ID
        and t1.p_typ = 'A'
    ;
------+------+------+------+------+------+------+------+------+------+---
       P1ID  P_TYP  P_TEXT            T2ID         P1ID  C_TYP  C_TEXT
------+------+------+------+------+------+------+------+------+------+---
          1  A      tab1 1              11            1  X      tab2 1 1
          5  A      tab1 5              51            5  X      tab2 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 2
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
------+------+------+------+------+------+------+------+------+------+---
   select  t1.* , t2.*
    from         session.tab1 as t1
      inner join session.tab2 as t2
        on  t1.P1ID = t2.P1ID
      where t1.p_typ = 'A'
    ;
------+------+------+------+------+------+------+------+------+------+---
       P1ID  P_TYP  P_TEXT            T2ID         P1ID  C_TYP  C_TEXT
------+------+------+------+------+------+------+------+------+------+---
          1  A      tab1 1              11            1  X      tab2 1 1
          5  A      tab1 5              51            5  X      tab2 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 2
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
------+------+------+------+------+------+------+------+------+------+---
   select  t1.* , t2.*
    from         session.tab1 as t1
      inner join session.tab2 as t2
        on  t1.P1ID = t2.P1ID
        and t2.c_typ = 'X'
    ;
------+------+------+------+------+------+------+------+------+------+---
       P1ID  P_TYP  P_TEXT            T2ID         P1ID  C_TYP  C_TEXT
------+------+------+------+------+------+------+------+------+------+---
          1  A      tab1 1              11            1  X      tab2 1 1
          4  B      tab1 4              41            4  X      tab2 4 1
          5  A      tab1 5              51            5  X      tab2 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 3
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
------+------+------+------+------+------+------+------+------+------+---
   select  t1.* , t2.*
    from         session.tab1 as t1
      inner join session.tab2   as t2
        on  t1.P1ID = t2.P1ID
      where t2.c_typ = 'X'
    ;
------+------+------+------+------+------+------+------+------+------+---
       P1ID  P_TYP  P_TEXT            T2ID         P1ID  C_TYP  C_TEXT
------+------+------+------+------+------+------+------+------+------+---
          1  A      tab1 1              11            1  X      tab2 1 1
          4  B      tab1 4              41            4  X      tab2 4 1
          5  A      tab1 5              51            5  X      tab2 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 3
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
```

Fazit: Beim Inner Join wirken ON-Klausel und WHERE äquivalant auf die Datenauswahl.

1.1.4.2 Left Outer Join

Tabelle 1

```
  select * from session.tab1 ;
------+------+------+------+-
P1ID  P_TYP  P_TEXT
------+------+------+------+--
1     A      tab1 1
2     B      tab1 2
4     B      tab1 4
5     A      tab1 5
DSNE610I NUMBER OF ROWS DISPLAYED IS 4
```

Tabelle 2

```
  select * from session.tab2 ;
------+------+------+------+------+------+-----
T2ID         P1ID  C_TYP  C_TEXT
------+------+------+------+------+------+-----
11            1    X      tab2 1 1
12  ----------     Y      tab2 NL 2
41            4    X      tab2 4 1
51            5    X      tab2 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 4
```

Der Left Outer Join führt zunächst logisch einen Inner Join durch. Dann werden alle nicht-übereinstimmenden Zeilen (hier: PID 2 der Parent-Tabelle) ebenfalls in die Result Table einbezogen.
Da für diesen PK keine Zeilen mit identischem FK existieren, werden die Spalten-Werte der Dependant Table mit NULL-Werten aufgefüllt. Dies bezieht sich auch auf Spalten, die per Definition nicht NULL-fähig sind, wie z.B. der PK CID.

Wird die ON-Klausel (neben dem PK-FK) erweitert, erweitert sich die Bedingung zur Erkennung der Gleichheit zweier Zeilen entsprechend. Siehe hierzu das erste Beispiel auf der rechten Seite.
Wie vorab ausgeführt, entscheidet nur die WHERE-Klausel, nicht aber die ON-Klausel über die bereitgestellte Daten-Menge.

```
-- left outer Join komplett
  select  t1.* , t2.*
    from         session.tab1 as t1
      left  join session.tab2   as t2
        on  t1.P1ID = t2.P1ID
    ;
------+------+------+------+------+------+------+------+------+------+------+---
       P1ID  P_TYP  P_TEXT             T2ID         P1ID  C_TYP  C_TEXT
------+------+------+------+------+------+------+------+------+------+------+---
          1  A      tab1 1               11            1  X      tab2 1 1
          2  B      tab1 2      -----------  -----------  -----  ----------
          4  B      tab1 4               41            4  X      tab2 4 1
          5  A      tab1 5               51            5  X      tab2 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 4
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
```

```
-- left  Join mit Filterung
   select  t1.* , t2.*
    from         session.tab1 as t1
      left  join session.tab2   as t2
        on  t1.P1ID = t2.P1ID
        and t1.p_typ = 'A'
    ;
------+------+------+------+------+------+------+------+------+------+---
       P1ID  P_TYP  P_TEXT             T2ID         P1ID  C_TYP  C_TEXT
------+------+------+------+------+------+------+------+------+------+---
          1  A      tab1 1               11            1  X      tab2 1 1
          2  B      tab1 2       -----------  -----------  -----  ----------
          4  B      tab1 4       -----------  -----------  -----  ----------
          5  A      tab1 5               51            5  X      tab2 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 4
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
------+------+------+------+------+------+------+------+------+------+---
   select  t1.* , t2.*
    from         session.tab1 as t1
      left  join session.tab2   as t2
        on  t1.P1ID = t2.P1ID
      where t1.p_typ = 'A'
    ;
------+------+------+------+------+------+------+------+------+------+---
       P1ID  P_TYP  P_TEXT             T2ID         P1ID  C_TYP  C_TEXT
------+------+------+------+------+------+------+------+------+------+---
          1  A      tab1 1               11            1  X      tab2 1 1
          5  A      tab1 5               51            5  X      tab2 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 2
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
------+------+------+------+------+------+------+------+------+------+---
   select  t1.* , t2.*
    from         session.tab1 as t1
      left  join session.tab2   as t2
        on  t1.P1ID = t2.P1ID
        and t2.c_typ = 'X'
    ;
------+------+------+------+------+------+------+------+------+------+---
       P1ID  P_TYP  P_TEXT             T2ID         P1ID  C_TYP  C_TEXT
------+------+------+------+------+------+------+------+------+------+---
          1  A      tab1 1               11            1  X      tab2 1 1
          2  B      tab1 2       -----------  -----------  -----  ----------
          4  B      tab1 4               41            4  X      tab2 4 1
          5  A      tab1 5               51            5  X      tab2 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 4
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
------+------+------+------+------+------+------+------+------+------+---
   select  t1.* , t2.*
    from         session.tab1 as t1
      left  join session.tab2   as t2
        on  t1.P1ID = t2.P1ID
      where t2.c_typ = 'X'
    ;
------+------+------+------+------+------+------+------+------+------+---
       P1ID  P_TYP  P_TEXT             T2ID         P1ID  C_TYP  C_TEXT
------+------+------+------+------+------+------+------+------+------+---
          1  A      tab1 1               11            1  X      tab2 1 1
          4  B      tab1 4               41            4  X      tab2 4 1
          5  A      tab1 5               51            5  X      tab2 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 3

DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
```

Fazit: Beim Outer Join wirken ON-Klausel und WHERE unterschiedlich auf die Datenauswahl.

1.1.4.3 Right Outer Join

Tabelle 1

```
  select * from session.tab1 ;
------+------+------+------+-
P1ID  P_TYP  P_TEXT
------+------+------+------+--
1     A      tab1 1
2     B      tab1 2
4     B      tab1 4
5     A      tab1 5
DSNE610I NUMBER OF ROWS DISPLAYED IS 4
```

Tabelle 2

```
  select * from session.tab2 ;
------+------+------+------+------+------+-----
T2ID         P1ID  C_TYP  C_TEXT
------+------+------+------+------+------+-----
11             1    X      tab2 1 1
12  -----------     Y      tab2 NL 2
41             4    X      tab2 4 1
51             5    X      tab2 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 4
```

Der Right Outer Join entspricht im Gegensatz zu den vorherigen Beispielen der Hierarchien nicht mehr dem Inner Join.

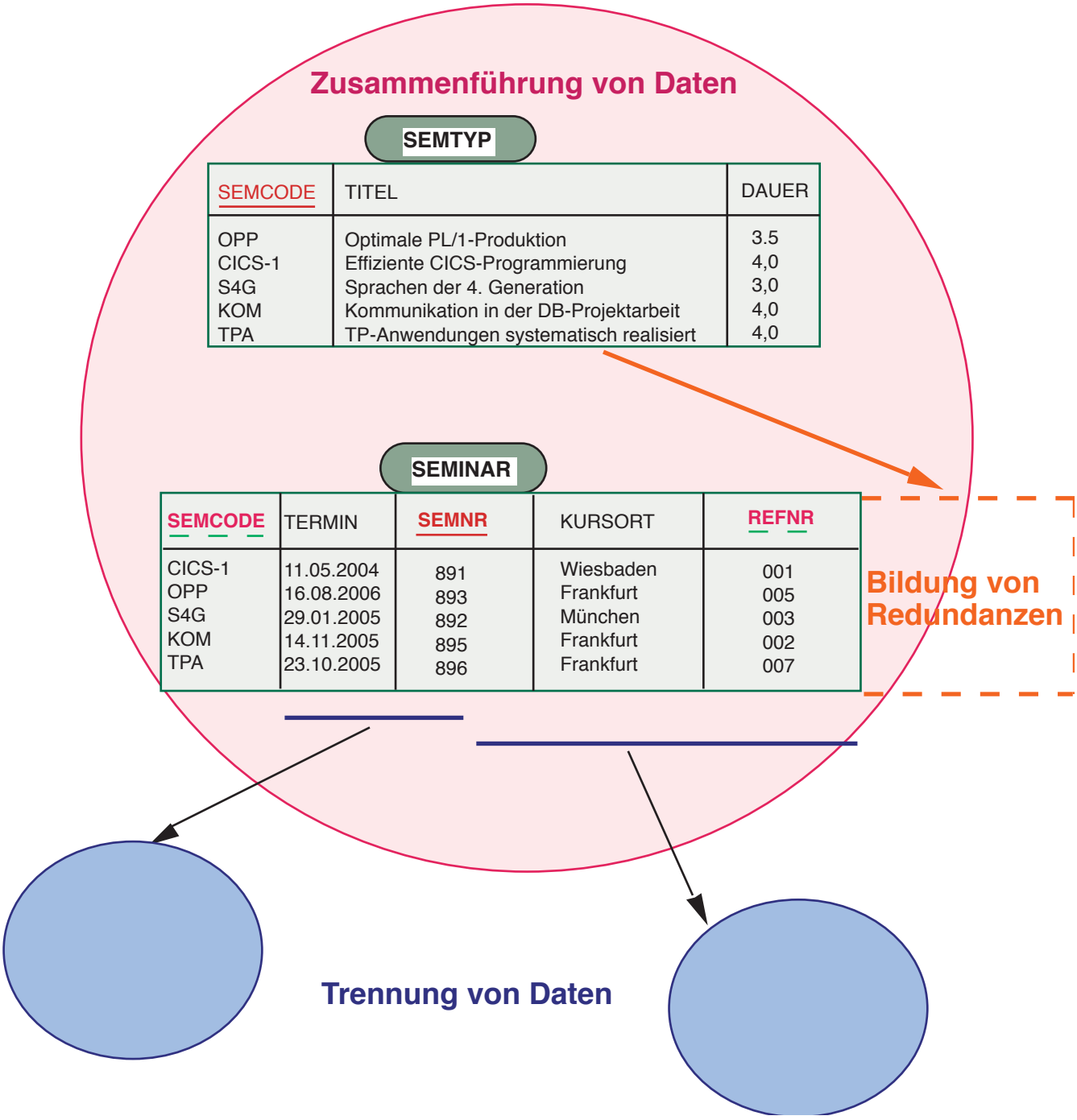
```
-- right outer Join komplett
  select  t1.* , t2.*
    from         session.tab1 as t1
      right join session.tab2   as t2
        on  t1.P1ID = t2.P1ID
    ;
------+------+------+------+------+------+------+------+------+------+------+---
       P1ID  P_TYP  P_TEXT             T2ID         P1ID  C_TYP  C_TEXT
------+------+------+------+------+------+------+------+------+------+------+---
          1  A      tab1 1               11            1  X      tab2 1 1
-----------  -----  ----------           12  -----------  Y      tab2 NL 2
          4  B      tab1 4               41            4  X      tab2 4 1
          5  A      tab1 5               51            5  X      tab2 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 4
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
```

```
-- right Join mit Filterung
   select  t1.* , t2.*
    from         session.tab1 as t1
      right join session.tab2   as t2
        on  t1.P1ID = t2.P1ID
        and t1.p_typ = 'A'
    ;
------+------+------+------+------+------+------+------+------+------+------+---
       P1ID  P_TYP  P_TEXT             T2ID         P1ID  C_TYP  C_TEXT
------+------+------+------+------+------+------+------+------+------+------+---
          1  A      tab1 1               11            1  X      tab2 1 1
-----------  -----  ----------           12  -----------  Y      tab2 NL 2
-----------  -----  ----------           41            4  X      tab2 4 1
          5  A      tab1 5               51            5  X      tab2 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 4
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
------+------+------+------+------+------+------+------+------+------+------+---
   select  t1.* , t2.*
    from         session.tab1 as t1
      right join session.tab2   as t2
        on  t1.P1ID = t2.P1ID
      where t1.p_typ = 'A'
    ;
------+------+------+------+------+------+------+------+------+------+------+---
       P1ID  P_TYP  P_TEXT             T2ID         P1ID  C_TYP  C_TEXT
------+------+------+------+------+------+------+------+------+------+------+---
          1  A      tab1 1               11            1  X      tab2 1 1
          5  A      tab1 5               51            5  X      tab2 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 2
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
------+------+------+------+------+------+------+------+------+------+------+---
   select  t1.* , t2.*
    from         session.tab1 as t1
      right join session.tab2   as t2
        on  t1.P1ID = t2.P1ID
        and t2.c_typ = 'X'
    ;
------+------+------+------+------+------+------+------+------+------+------+---
       P1ID  P_TYP  P_TEXT             T2ID         P1ID  C_TYP  C_TEXT
------+------+------+------+------+------+------+------+------+------+------+---
          1  A      tab1 1               11            1  X      tab2 1 1
-----------  -----  ----------           12  -----------  Y      tab2 NL 2
          4  B      tab1 4               41            4  X      tab2 4 1
          5  A      tab1 5               51            5  X      tab2 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 4
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
------+------+------+------+------+------+------+------+------+------+------+---
   select  t1.* , t2.*
    from         session.tab1 as t1
      right join session.tab2   as t2
        on  t1.P1ID = t2.P1ID
      where t2.c_typ = 'X'
    ;
------+------+------+------+------+------+------+------+------+------+------+---
       P1ID  P_TYP  P_TEXT             T2ID         P1ID  C_TYP  C_TEXT
------+------+------+------+------+------+------+------+------+------+------+---
          1  A      tab1 1               11            1  X      tab2 1 1
          4  B      tab1 4               41            4  X      tab2 4 1
          5  A      tab1 5               51            5  X      tab2 5 1
DSNE610I NUMBER OF ROWS DISPLAYED IS 3

DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
----+----+----+----+----+----+----+----+----+----+----+----+----+-+--------
```

Fazit: Beim Outer Join wirken ON-Klausel und WHERE unterschiedlich auf die Datenauswahl.

1.1.5 Einflüsse des physischen internen Daten-Modells
1.1.5.1 De-Normalisierungs-Varianten



Jede Strukturveränderung im Vergleich zum logischen Modell führt zu SQL-Funktions-Konsequenzen!

Werden relevante Datensichten häufig benötigt, können Überlegungen einsetzen, wie der Daten- Beschaffungs- und -Entsorgungs-Aufwand optimaler gestaltet werden kann.

Dabei können auch Struktur-Veränderungen in die Überlegungen einbezogen werden.
Folgende grundsätzliche Formen existieren:

1 **Zusammenführung häufig gemeinsam benötigter Daten**
Zusammenfassung der Daten separater Tabellen in einer einzigen Tabelle.

Argumente für diese Vorgehensweise:
- Weniger physische Objekte.
- Abbau von Join-Aufwendungen.
- Abbau von Manipulations-Aufwendungen.

2 **Trennung häufig benötigter Daten von weniger häufig benötigter Daten**

Aufteilung der Daten einer in zwei oder mehr Tabellen:
- Struktur-Trennung (Spalten werden aufgeteilt).
- Verteilung der Dateninhalte (Zeilen-Trennung).

Argumente für diese Vorgehensweise:
- Verteilung der Daten an die Stellen, an denen sie überwiegend benötigt werden.
- Aufteilung häufig benötigter Daten von selten benötigten Daten.
- Trennung von Daten mit unterschiedlichen Charakteristiken.

3 **Bildung von Redundanzen**

Häufig benötigte Daten aus einer oder mehreren Tabellen werden redundant in anderen Tabellen aufgenommen.

Argumente für diese Vorgehensweise:
- Abbau von Join-Aufwendungen.
- Abbau von aufwändigen Funktions-Durchläufen, z.B. bei Aggregations-Werten.

1.1.6 Trennung der Daten (Verteilung in mehrere Tabellen)

1.1.6.1 Trennung in Aktuell, Historisch und Zukunft

Tabelle Aktuell

```
select * from session.aktuell ;
  +-------------------------------------------------+
  |      PID       |   GILTAB   | TYP |      TEXT       |
  +-------------------------------------------------+
1_|              1 | 2008-01-01 | AA  | aktuell 1       |
2_|              2 | 2008-03-15 | AA  | aktuell 2       |
3_|              3 | 2007-11-15 | AA  | aktuell 3       |
4_|              4 | 2000-03-17 | BB  | aktuell 4       |
5_|              5 | 2006-05-15 | AA  | aktuell 5       |
+-------------------------------------------------+
 SUCCESSFUL RETRIEVAL OF        5 ROW(S)
```

Tabelle Historie

```
select * from session.historie ;
  +-------------------------------------------------+
  |      PID       |   GILTAB   | TYP |      TEXT       |
  +-------------------------------------------------+
1_|              1 | 2006-01-01 | AA  | historie 1a     |
2_|              1 | 2007-07-21 | AA  | historie 1b     |
3_|              2 | 2000-01-01 | BB  | historie 2a     |
4_|              2 | 2006-06-30 | AA  | historie 2b     |
5_|              2 | 2007-12-06 | AA  | historie 2c     |
6_|              5 | 2000-01-01 | BB  | historie 5      |
  +-------------------------------------------------+
 SUCCESSFUL RETRIEVAL OF        6 ROW(S)
```

Tabelle Zukunft

```
select * from session.zukunft   ;
  +-------------------------------------------------+
  |      PID       |   GILTAB   | TYP |      TEXT       |
  +-------------------------------------------------+
1_|              1 | 2009-01-01 | AA  | zukunft 1a      |
2_|              1 | 2009-03-15 | AA  | zukunft 1b      |
3_|              5 | 2009-05-15 | AA  | zukunft 5       |
4_|              6 | 2010-10-01 | AA  | zukunft 6       |
  +-------------------------------------------------+
 SUCCESSFUL RETRIEVAL OF        4 ROW(S)
```

1.1.6.1.1 Inner- und Outer Join

```
   -- Inner Join komplett
    select  a.* , h.* , z.*
      from         session.aktuell  as a
        inner join session.historie as h
             on    a.pid = h.pid
        inner join session.zukunft  as z
             on    a.pid = z.pid   ;
```

	PID	GILTAB	TYP	TEXT	PID	GILTAB	TYP	TEXT
1_	1	2008-01-01	AA	aktuell 1	1	2006-01-01	AA	historie 1a
2_	1	2008-01-01	AA	aktuell 1	1	2007-07-21	AA	historie 1b
3_	1	2008-01-01	AA	aktuell 1	1	2006-01-01	AA	historie 1a
4_	1	2008-01-01	AA	aktuell 1	1	2007-07-21	AA	historie 1b
5_	5	2006-05-15	AA	aktuell 5	5	2000-01-01	BB	historie 5

	PID	GILTAB	TYP	TEXT
1_	1	2009-01-01	AA	zukunft 1a
2_	1	2009-01-01	AA	zukunft 1a
3_	1	2009-03-15	AA	zukunft 1b
4_	1	2009-03-15	AA	zukunft 1b
5_	5	2009-05-15	AA	zukunft 5

```
0SUCCESSFUL RETRIEVAL OF        5 ROW(S)
  -- full outer Join komplett
    select  a.* , h.* , z.*
      from         session.aktuell  as a
        full  join session.historie as h
             on    a.pid = h.pid
        full  join session.zukunft  as z
             on    a.pid = z.pid   ;
```

	PID	GILTAB	TYP	TEXT	PID	GILTAB	TYP	TEXT
1_	1	2008-01-01	AA	aktuell 1	1	2006-01-01	AA	historie 1a
2_	1	2008-01-01	AA	aktuell 1	1	2006-01-01	AA	historie 1a
3_	1	2008-01-01	AA	aktuell 1	1	2007-07-21	AA	historie 1b
4_	1	2008-01-01	AA	aktuell 1	1	2007-07-21	AA	historie 1b
5_	2	2008-03-15	AA	aktuell 2	2	2006-06-30	AA	historie 2b
6_	2	2008-03-15	AA	aktuell 2	2	2007-12-06	AA	historie 2c
7_	2	2008-03-15	AA	aktuell 2	2	2000-01-01	BB	historie 2a
8_	3	2007-11-15	AA	aktuell 3	?	?	?	?
9_	4	2000-03-17	BB	aktuell 4	?	?	?	?
10_	5	2006-05-15	AA	aktuell 5	5	2000-01-01	BB	historie 5
11_	?	?	?	?	?	?	?	?

	PID	GILTAB	TYP	TEXT
1_	1	2009-03-15	AA	zukunft 1b
2_	1	2009-01-01	AA	zukunft 1a
3_	1	2009-03-15	AA	zukunft 1b
4_	1	2009-01-01	AA	zukunft 1a
5_	?	?	?	?
6_	?	?	?	?
7_	?	?	?	?
8_	?	?	?	?
9_	?	?	?	?
10_	5	2009-05-15	AA	zukunft 5
11_	6	2010-10-01	AA	zukunft 6

```
0SUCCESSFUL RETRIEVAL OF       11 ROW(S)
```

1.1.6.1.2 UNION

Mit UNION können die Result Tables zweier SELECT-Statements zu einer gemeinsamen Result Table verknüpft werden.

Dabei werden die einzelnen SELECT-Statements separat ausgeführt und deren Ergebnisse werden in temporären Result Tables zwischengespeichert.
Die endgültige Result Table wird in Abhängigkeit vom UNION-Typ erzeugt:

- **UNION** Zusammenmischen und eliminieren von Redundanzen (DISTINCT ist implizit).
- **UNION ALL** Zusammenmischen und erhalten der Redundanzen.

```
-- UNION
  select  a.*
    from         session.aktuell  as a
  union     all
  select  h.*
    from         session.historie as h
  union     all
  select  z.*
    from         session.zukunft  as z
  ;
```

```
                          +-------------------------------------------------+
                          |      PID       |   GILTAB   | TYP |      TEXT       |
                          +-------------------------------------------------+
                        1_|              1 | 2008-01-01 | AA  | aktuell 1       |
                        2_|              2 | 2008-03-15 | AA  | aktuell 2       |
                        3_|              3 | 2007-11-15 | AA  | aktuell 3       |
                        4_|              4 | 2000-03-17 | BB  | aktuell 4       |
                        5_|              5 | 2006-05-15 | AA  | aktuell 5       |
                        6_|              1 | 2006-01-01 | AA  | historie 1a     |
                        7_|              1 | 2007-07-21 | AA  | historie 1b     |
                        8_|              2 | 2000-01-01 | BB  | historie 2a     |
                        9_|              2 | 2006-06-30 | AA  | historie 2b     |
                       10_|              2 | 2007-12-06 | AA  | historie 2c     |
                       11_|              5 | 2000-01-01 | BB  | historie 5      |
                       12_|              1 | 2009-01-01 | AA  | zukunft 1a      |
                       13_|              1 | 2009-03-15 | AA  | zukunft 1b      |
                       14_|              5 | 2009-05-15 | AA  | zukunft 5       |
                       15_|              6 | 2010-10-01 | AA  | zukunft 6       |
                          +-------------------------------------------------+
0SUCCESSFUL RETRIEVAL OF       15 ROW(S)
1PAGE    1
```

```
-- UNION mit Sort
 select  a.*
   from         session.aktuell  as a
 union     all
 select  h.*
   from         session.historie as h
 union     all
 select  z.*
   from         session.zukunft  as z
 order by pid , giltab
 ;
```

	PID	GILTAB	TYP	TEXT
1_	1	2006-01-01	AA	historie 1a
2_	1	2007-07-21	AA	historie 1b
3_	1	2008-01-01	AA	aktuell 1
4_	1	2009-01-01	AA	zukunft 1a
5_	1	2009-03-15	AA	zukunft 1b
6_	2	2000-01-01	BB	historie 2a
7_	2	2006-06-30	AA	historie 2b
8_	2	2007-12-06	AA	historie 2c
9_	2	2008-03-15	AA	aktuell 2
10_	3	2007-11-15	AA	aktuell 3
11_	4	2000-03-17	BB	aktuell 4
12_	5	2000-01-01	BB	historie 5
13_	5	2006-05-15	AA	aktuell 5
14_	5	2009-05-15	AA	zukunft 5
15_	6	2010-10-01	AA	zukunft 6

```
0SUCCESSFUL RETRIEVAL OF       15 ROW(S)
```

1.1.6.1.3 INTERSECT

Das Ergebnis eines INTERSECT-Operators enthält alle Zeilen, die in beiden Result Tables auftreten.

Der Vergleich erfolgt auf der Ebene der kompletten Result-Table-Zeile. Daher ist die Spalten-Auswahl wichtig. Unterscheiden sich bestimmte Spalten-Inhalte immer, bringt der Operator kein sinnvolles Ergebnis. Daher wurde in der zweiten Query eine Reduzierung auf eine aussagefähige Spalte vorgenommen.

```
    -- INTERSECT komplett
 ***INPUT STATEMENT:
    select  a.*
      from         session.aktuell  as a
    intersect all
    select  h.*
      from         session.historie as h
    intersect all
    select  z.*
      from         session.zukunft  as z
    ;
0SUCCESSFUL RETRIEVAL OF        0 ROW(S)



   -- INTERSECT PID , d.h. alle PIDs, die in allen Tabellen auftreten
 ***INPUT STATEMENT:
    select  a.pid
      from         session.aktuell  as a
    intersect all
    select  h.pid
      from         session.historie as h
    intersect all
    select  z.pid
      from         session.zukunft  as z
    ;
                                        +-----------------+
                                        |       PID       |
                                        +-----------------+
                                      1_|               1 |
                                      2_|               5 |
                                        +-----------------+
0SUCCESSFUL RETRIEVAL OF        2 ROW(S)

0SUCCESSFUL RETRIEVAL OF        4 ROW(S)
1PAGE    1
   -- INTERSECT    , bezogen auf TYP und PID
 ***INPUT STATEMENT:
    select  typ , pid     from         session.zukunft  as z
     intersect all
    select  typ , pid
      from         session.aktuell  as a
    intersect all
    select  typ , pid
      from         session.historie as h
    ;
                                        +-----------------------+
                                        | TYP |       PID       |
                                        +-----------------------+
                                      1_| AA  |               1 |
                                        +-----------------------+
0SUCCESSFUL RETRIEVAL OF        1 ROW(S)
```

1.1.6.1.4 EXCEPT

Das Ergebnis eines EXCEPT-Operators enthält alle Zeilen, die nicht in der ersten, aber nicht in der zweiten Result Table auftreten.

Hier spielen folgende Faktoren eine wichtige Rolle:

- **Positionierung der beiden Tabellen innerhalb des SQL-Statements (erste und zweite Tabelle)**
- **Spezifikation der beteiligten Spalten (Regeln analog INTERSECT vorab).**

```
 -- EXCEPT    komplett
***INPUT STATEMENT:
   select  a.*
     from         session.aktuell  as a
   except    all
   select  h.*
     from         session.historie as h
   except    all
   select  z.*
     from         session.zukunft  as z
   ;
```

	PID	GILTAB	TYP	TEXT
1_	1	2008-01-01	AA	aktuell 1
2_	2	2008-03-15	AA	aktuell 2
3_	3	2007-11-15	AA	aktuell 3
4_	4	2000-03-17	BB	aktuell 4
5_	5	2006-05-15	AA	aktuell 5

```
0SUCCESSFUL RETRIEVAL OF        5 ROW(S)

  -- EXCEPT   PID , d.h. PIDs, die nur in der aktuellen Tab auftreten
***INPUT STATEMENT:
   select  a.pid
     from         session.aktuell  as a
   except    all
   select  h.pid
     from         session.historie as h
   except    all
   select  z.pid
     from         session.zukunft  as z
   ;
```

	PID
1_	3
2_	4

```
0SUCCESSFUL RETRIEVAL OF        2 ROW(S)
```

```
 -- EXCEPT   PID , d.h. PIDs, die nur in der histor.   Tab auftreten
***INPUT STATEMENT:
   select  h.pid
     from         session.historie as h
   except    all
   select  a.pid
     from         session.aktuell  as a
   except    all
   select  z.pid
     from         session.zukunft  as z
   ;
                                      +-----------------+
                                      |      PID        |
                                      +-----------------+
                                    1_|               2 |
                                    2_|               2 |
                                      +-----------------+
0SUCCESSFUL RETRIEVAL OF        2 ROW(S)


  -- EXCEPT   PID , d.h. PIDs, die nur in der zukuenft. Tab auftreten

   select  z.pid    from         session.zukunft  as z
    except    all
   select  h.pid    from         session.historie as h
    except    all
   select  a.pid    from         session.aktuell  as a  ;
                                        +-----------------+
                                        |      PID        |
                                        +-----------------+
                                      1_|               6 |
                                        +-----------------+
 SUCCESSFUL RETRIEVAL OF        1 ROW(S)


  -- EXCEPT   PID , wie vorab, andere Reihenfolge

   select  z.pid    from         session.zukunft  as z
    except    all
   select  a.pid    from         session.aktuell  as a
    except    all
   select  h.pid    from         session.historie as h   ;
                                        +-----------------+
                                        |      PID        |
                                        +-----------------+
                                      1_|               6 |
                                        +-----------------+
 SUCCESSFUL RETRIEVAL OF        1 ROW(S)

  -- EXCEPT   PID , wie vorab, mit Spalten-Erweiterung

   select  z.pid , giltab   from         session.zukunft  as z
    except    all
   select  a.pid , giltab   from         session.aktuell  as a
    except    all
   select  h.pid , giltab   from         session.historie as h   ;
                                        +-------------------------------+
                                        |      PID        |   GILTAB    |
                                        +-------------------------------+
                                      1_|               1 | 2009-01-01  |
                                      2_|               1 | 2009-03-15  |
                                      3_|               5 | 2009-05-15  |
                                      4_|               6 | 2010-10-01  |
                                        +-------------------------------+
```

```
-- EXCEPT       , bezogen auf TYP und PID
    select  typ , pid
      from         session.zukunft  as z
    except    all
    select  typ , pid
      from         session.aktuell  as a
    except    all
    select  typ , pid
      from         session.historie as h
    ;
```

	TYP	PID
1_	AA	6

```
 SUCCESSFUL RETRIEVAL OF        1 ROW(S)
```

1.1.7 Daten-Trennung und virtuelle Zusammenführung

1.1.7.1 Trennung in Produktiv und Individuell (überlagernd)

Tabelle Produktion

```
SELECT *
      FROM SESSION.PROD ;
  +-----------------------------------------+
  !    KEY    !     C1     !     C2     ! STATUS !
  +-----------------------------------------+
1_!         1 ! P1A        ! P1B        ! A      !
2_!         2 ! P2A        ! P2B        ! A      !
3_!         4 ! P4A        ! P4B        ! A      !
4_!         5 ! P5A        ! P5B        ! A      !
  +-----------------------------------------+
```

Tabelle individuelle Änderungen TEST

```
 SELECT *
      FROM SESSION.TEST ;
  +-----------------------------------------+
  !    KEY    !     C1     !     C2     ! STATUS !
  +-----------------------------------------+
1_!         2 ! P2A        ! D2B        ! A      !
2_!         1 ! GEL        ! GEL        ! D      !
3_!         6 ! D6A        ! D6B        ! A      !
  +-----------------------------------------+
```

Tabelle **Produktion** enthält die Basis-Daten, die als Grundlage wirken.
Diese Tabelle enthält die Spalten:

- Key PK
- C1 Beliebige Spalte 1
- C2 Beliebige Spalte 2
- STATUS A = Aktiv

Mit der Tabelle **TEST** können individuelle Überlagerungen vorgegeben werden, die auf der Basis der Tabelle Produktion entstanden sind. Steht hier keine Zeile für einen KEY zur Verfügung, zieht die Tabelle PRODUKTION, ansonsten die Tabelle TEST. Ist hier eine Zeile mit STATUS D (Delete), darf die Basis-Datenzeile nicht ausgegeben werden.
Diese Tabelle enthält die Spalten:

- Key PK
- C1 Beliebige Spalte 1
- C2 Beliebige Spalte 2
- STATUS A = Aktiv D = Delete

1.1.7.1.1 UNION und UNION ALL

```
  INPUT STATEMENT:
   SELECT P.*, 'PROD' as Herkunft
      FROM SESSION.PROD P
   UNION ALL                                    -- UNION bringt dasselbe Ergebnis
   SELECT T.*, 'TEST ' as Herkunft
      FROM SESSION.TEST T
      ORDER BY KEY ;
  +-------------------------------------------------------+
  !    KEY    !     C1     !     C2     ! STATUS ! HERKUNFT !
  +-------------------------------------------------------+
1_!         1 ! GEL        ! GEL        ! D      ! TEST     !
2_!         1 ! P1A        ! P1B        ! A      ! PROD     !
3_!         2 ! P2A        ! D2B        ! A      ! TEST     !
4_!         2 ! P2A        ! P2B        ! A      ! PROD     !
5_!         4 ! P4A        ! P4B        ! A      ! PROD     !
6_!         5 ! P5A        ! P5B        ! A      ! PROD     !
7_!         6 ! D6A        ! D6B        ! A      ! TEST     !
  +-------------------------------------------------------+
SUCCESSFUL RETRIEVAL OF        7 ROW(S)
```

1.1.7.1.2 INTERSECT und INTERSECT ALL

```
    SELECT P.*, 'PROD' as Herkunft
      FROM SESSION.PROD P
   INTERSECT ALL                                -- INTERSECT bringt dasselbe Ergebnis
   SELECT T.*, 'TEST ' as Herkunft
      FROM SESSION.TEST T
      ORDER BY KEY ;
SUCCESSFUL RETRIEVAL OF        0 ROW(S)-------------+
```

1.1.7.1.3 EXCEPT und EXCEPT ALL

```
   SELECT P.*, 'PROD' as Herkunft
      FROM SESSION.PROD P
   EXCEPT    ALL                                -- EXCEPT bringt dasselbe Ergebnis
   SELECT T.*, 'TEST ' as Herkunft
      FROM SESSION.TEST T
      ORDER BY KEY ;
  +-------------------------------------------------------+
  !    KEY    !     C1     !     C2     ! STATUS ! HERKUNFT !
  +-------------------------------------------------------+
1_!         1 ! P1A        ! P1B        ! A      ! PROD     !
2_!         2 ! P2A        ! P2B        ! A      ! PROD     !
3_!         4 ! P4A        ! P4B        ! A      ! PROD     !
4_!         5 ! P5A        ! P5B        ! A      ! PROD     !
  +-------------------------------------------------------+
SUCCESSFUL RETRIEVAL OF        4 ROW(S)

   SELECT T.*, 'TEST ' as Herkunft
      FROM SESSION.TEST T
   EXCEPT    ALL                                -- EXCEPT bringt dasselbe Ergebnis
   SELECT P.*, 'PROD' as Herkunft
      FROM SESSION.PROD P
      ORDER BY KEY ;
  +-------------------------------------------------------+
  !    KEY    !     C1     !     C2     ! STATUS ! HERKUNFT !
  +-------------------------------------------------------+
1_!         1 ! GEL        ! GEL        ! D      ! TEST     !
2_!         2 ! P2A        ! D2B        ! A      ! TEST     !
3_!         6 ! D6A        ! D6B        ! A      ! TEST     !
  +-------------------------------------------------------+
SUCCESSFUL RETRIEVAL OF        3 ROW(S)
```

1.1.7.1.4 Inner Join

```
   SELECT P.KEY                 AS KEY
        ,  P.C1                 AS PROD_C1
        ,  P.C2                 AS PROD_C2
        ,  P.STATUS             AS PROD_STATUS
        ,  T.C1                 AS TEST_C1
        ,  T.C2                 AS TEST_C2
        ,  T.STATUS             AS TEST_STATUS
     FROM SESSION.PROD AS P
        INNER       JOIN SESSION.TEST AS T
       ON  P.KEY = T.KEY
    ORDER BY KEY ;
  +-------------------------------------------------------------------------------------------+
  !    KEY    !  PROD_C1   !  PROD_C2   ! PROD_STATUS !  TEST_C1   !  TEST_C2  ! TEST_STATUS !
  +-------------------------------------------------------------------------------------------+
1_!         1 ! P1A        ! P1B        ! A           ! GEL        ! GEL       ! D           !
2_!         2 ! P2A        ! P2B        ! A           ! P2A        ! D2B       ! A           !
  +-------------------------------------------------------------------------------------------+
 SUCCESSFUL RETRIEVAL OF        2 ROW(S)
```

1.1.7.1.5 Left und Right Outer Join, mit UNION verknüpft

```
   SELECT P.KEY                AS KEY
        ,  P.C1                 AS C1
        ,  P.C2                 AS C2
        ,  P.STATUS             AS STATUS
     FROM SESSION.PROD AS P
        LEFT OUTER  JOIN SESSION.TEST AS T
       ON  P.KEY = T.KEY
   UNION
    SELECT T.KEY                AS KEY
        ,  T.C1                 AS C1
        ,  T.C2                 AS C2
        ,  T.STATUS             AS STATUS
     FROM SESSION.PROD AS P
        RIGHT OUTER JOIN SESSION.TEST AS T
       ON  P.KEY = T.KEY
    WHERE
          T.STATUS = 'A'
   ORDER BY KEY ;
  +-----------------------------------------------+
  !    KEY    !     C1     !     C2     ! STATUS !
  +-----------------------------------------------+
 1_!         1 ! P1A        ! P1B        ! A      !
 2_!         2 ! P2A        ! D2B        ! A      !
 3_!         2 ! P2A        ! P2B        ! A      !
 4_!         4 ! P4A        ! P4B        ! A      !
 5_!         5 ! P5A        ! P5B        ! A      !
 6_!         6 ! D6A        ! D6B        ! A      !
  +-----------------------------------------------+
SUCCESSFUL RETRIEVAL OF        6 ROW(S)
```

1.1.7.1.6 Full Outer Join

```
   SELECT COALESCE ( T.KEY , P.KEY ) AS KEY
       ,  COALESCE ( T.C1 , P.C1 )    AS C1
       ,  COALESCE ( T.C2 , P.C2 )    AS C2
       ,  COALESCE ( T.STATUS , P.STATUS)    AS STATUS
       ,  CASE
              WHEN T.KEY IS NULL THEN 'PROD'
                                  ELSE 'TEST' END AS HERKUNFT
     FROM SESSION.PROD AS P
        FULL OUTER  JOIN SESSION.TEST AS T
       ON  P.KEY = T.KEY
    ORDER BY KEY ;
  +-------------------------------------------------------+
  !    KEY    !     C1     !     C2     ! STATUS ! HERKUNFT !
  +-------------------------------------------------------+
1_!         1 ! GEL        ! GEL        ! D      ! TEST     !
2_!         2 ! P2A        ! D2B        ! A      ! TEST     !
3_!         4 ! P4A        ! P4B        ! A      ! PROD     !
4_!         5 ! P5A        ! P5B        ! A      ! PROD     !
5_!         6 ! D6A        ! D6B        ! A      ! TEST     !
  +-------------------------------------------------------+
SUCCESSFUL RETRIEVAL OF        5 ROW(S)

   SELECT COALESCE ( T.KEY , P.KEY ) AS KEY
       ,  COALESCE ( T.C1 , P.C1 )    AS C1
       ,  COALESCE ( T.C2 , P.C2 )    AS C2
       ,  COALESCE ( T.STATUS , P.STATUS)    AS STATUS
       ,  CASE
              WHEN T.KEY IS NULL THEN 'PROD'
                                  ELSE 'TEST' END AS HERKUNFT
     FROM SESSION.PROD AS P
        FULL OUTER  JOIN SESSION.TEST AS T
       ON  P.KEY = T.KEY
    WHERE
         T.STATUS = 'A'
   ORDER BY KEY ;
  +-------------------------------------------------------+
  !    KEY    !     C1     !     C2     ! STATUS ! HERKUNFT !
  +-------------------------------------------------------+
1_!         2 ! P2A        ! D2B        ! A      ! TEST     !
2_!         6 ! D6A        ! D6B        ! A      ! TEST     !
  +-------------------------------------------------------+
SUCCESSFUL RETRIEVAL OF        2 ROW(S)
```

```
    SELECT COALESCE ( T.KEY , P.KEY ) AS KEY
        ,  COALESCE ( T.C1 , P.C1 )    AS C1
        ,  COALESCE ( T.C2 , P.C2 )    AS C2
        ,  COALESCE ( T.STATUS , P.STATUS)    AS STATUS
        ,  CASE  WHEN T.KEY IS NULL THEN 'PROD'
                                    ELSE 'TEST' END AS HERKUNFT
      FROM SESSION.PROD AS P
         FULL OUTER  JOIN SESSION.TEST AS T
        ON  P.KEY = T.KEY
        AND T.STATUS = 'A'
    ORDER BY KEY ;
 SQLERROR ON   SELECT    COMMAND, PREPARE   FUNCTION
  RESULT OF SQL STATEMENT:
  DSNT408I SQLCODE = -338, ERROR:  AN ON CLAUSE IS INVALID
  DSNT418I SQLSTATE   = 42972 SQLSTATE RETURN CODE

   SELECT COALESCE ( T.KEY , P.KEY ) AS KEY
        ,  COALESCE ( T.C1 , P.C1 )    AS C1
        ,  COALESCE ( T.C2 , P.C2 )    AS C2
        ,  COALESCE ( T.STATUS , P.STATUS)    AS STATUS
        ,  CASE WHEN T.KEY IS NULL THEN 'PROD'
                                    ELSE 'TEST' END AS HERKUNFT
      FROM SESSION.PROD AS P
         FULL OUTER  JOIN SESSION.TEST AS T
        ON  P.KEY = T.KEY
        AND T.STATUS = P.STATUS
    ORDER BY KEY ;
   +-----------------------------------------------------------+
   !    KEY    !     C1     !     C2     ! STATUS ! HERKUNFT !
   +-----------------------------------------------------------+
 1_!         1 ! GEL        ! GEL        ! D      ! TEST     !
 2_!         1 ! P1A        ! P1B        ! A      ! PROD     !
 3_!         2 ! P2A        ! D2B        ! A      ! TEST     !
 4_!         4 ! P4A        ! P4B        ! A      ! PROD     !
 5_!         5 ! P5A        ! P5B        ! A      ! PROD     !
 6_!         6 ! D6A        ! D6B        ! A      ! TEST     !
   +-----------------------------------------------------------+
 SUCCESSFUL RETRIEVAL OF        6 ROW(S)


WITH TEMP1 AS (
    SELECT COALESCE ( T.KEY , P.KEY ) AS KEY
        ,  COALESCE ( T.C1 , P.C1 )    AS C1
        ,  COALESCE ( T.C2 , P.C2 )    AS C2
        ,  COALESCE ( T.STATUS , P.STATUS)    AS STATUS
        ,  CASE WHEN T.KEY IS NULL THEN 'PROD'
                                    ELSE 'TEST' END AS HERKUNFT
      FROM SESSION.PROD AS P
         FULL OUTER  JOIN SESSION.TEST AS T
        ON  P.KEY = T.KEY
   )
   SELECT * FROM TEMP1
     WHERE  STATUS = 'A'
    ORDER BY KEY  ;
   +-----------------------------------------------------------+
   !    KEY    !     C1     !     C2     ! STATUS ! HERKUNFT !
   +-----------------------------------------------------------+
 1_!         2 ! P2A        ! D2B        ! A      ! TEST     !
 2_!         4 ! P4A        ! P4B        ! A      ! PROD     !
 3_!         5 ! P5A        ! P5B        ! A      ! PROD     !
 4_!         6 ! D6A        ! D6B        ! A      ! TEST     !
   +-----------------------------------------------------------+
 SUCCESSFUL RETRIEVAL OF        4 ROW(S)
```