Benutzerdefinierte Datenexporte (BDEs)

- Erfassen und Aufrufen von BDEs
- HLR: Umfuhren
- Vollständige Liste vorhandener BDEs

Erfassen und Aufrufen von BDEs

Was sind benutzerdefinierte Datenexporte (BDE)

Ein benutzerdefinierter Datenexport (BDE) ist ein im ERP angelegter Bericht, der aufgrund einer hinterlegten SQL-Abfrage Daten ausgibt. Es können beliebig viele BDE angelegt werden.

Erfassen eines benutzerdefinierten Datenexports

Um einen BDE anzulegen, wählen Sie im ERP "System>Benutzerdefinierter Datenexport - Erstellen/Anpassen". Dort können Sie bestehende BDEs anpassen oder einen neuen über "Erfassen" anlegen.



Beim Erstellen müssen Sie folgende Felder ausfüllen:

- **Name:** Einen möglichst Sprechenden Namen wählen
- **Benötigtes Zugriffsrecht:** Wählen Sie die entsprechende Benutzergruppe aus
- **Beschreibung:** Die Beschreibung sollte alle benötigten Eingabe- und Ausgabefelder beinhalten
- **SQL-Abfrage:** Ein gültiger "Select"-Befehl der die nötigen Felder ausgibt.

Außerdem können im SQL-Befehl auch Abfrageparameter gesetzt werden. Diese müssen wie folgt im SQL-Befehl vorkommen: <%Variablenname%>

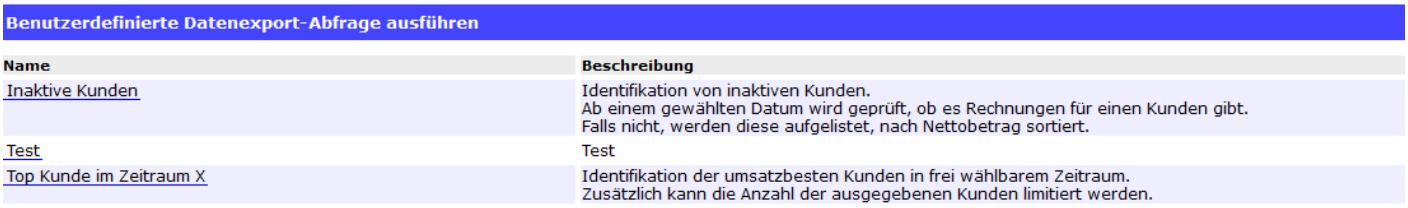
Auf der zweiten Seite der Erfassung können die Abfrageparameter festgelegt werden:



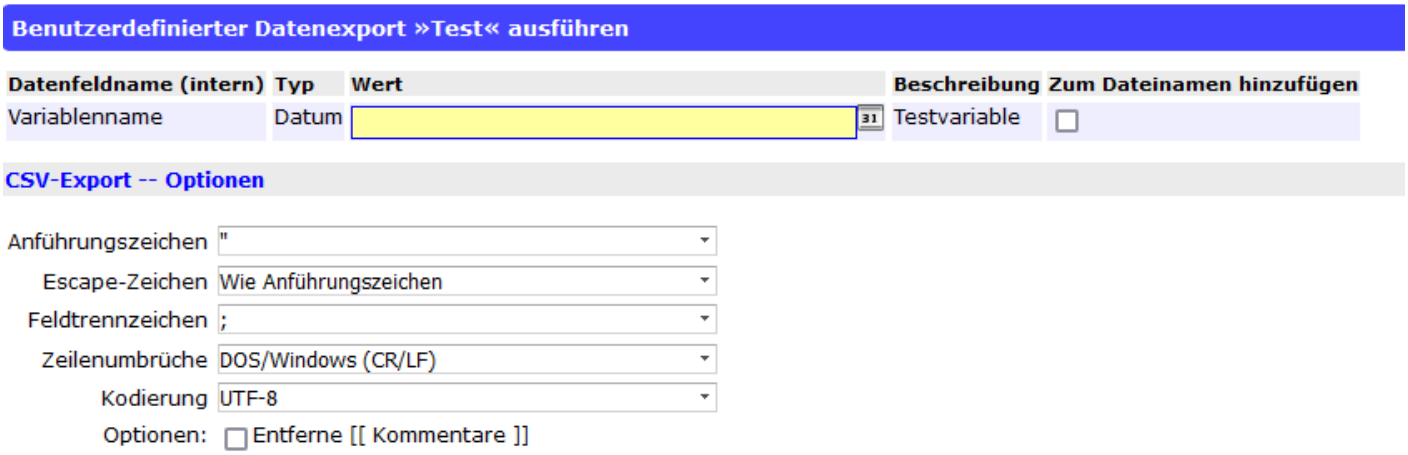
- **Datenfeldname (intern):** Ist der Name des Abfrageparameters sowohl in der SQL-Abfrage als auch in der Eingabemaske
- **Typ:** Wählen Sie die Art des Abfrageparameters aus (Beispiel: Datum)
- **Beschreibung:** Beschreiben Sie die gewünschte Art der Eingabe (Beispiel: Datum von)
- **Standardwert:** Wählen Sie, falls gewollt, einen Standardwert aus. Je nach Auswahl können Sie in dem Textfeld rechts einen Wert hinterlegen. (Beispiel: Fester Wert > 01.01.2025)

Aufrufen eines BDE

Ihre benutzerdefinierten Datenexporte können Sie unter "Berichte>Benutzerdefinierter Datenexport" anzeigen.



Hier sind beispielhaft ein paar BDEs zu sehen. Beim Ausführen kommen Sie zuerst auf ein Eingabeformular, in dem Sie die Eingabeparameter setzen und den "CSV-Export" anpassen können.



- Über den Button "**Vorschau**" können Sie sich dann den Bericht anzeigen lassen.
- In der "**Vorschau**" können Sie dann über den Button "**Export**" die Tabelle als CSV exportieren.

- Auf der folgende Seite des Kapitels finden Sie Beispiele für BDEs, die Sie bei sich übernehmen können.

HLR: Umfuhren

Für den Waren-Transport zwischen Firmierung dienen die „Umfuhren“ als folgende Funktion:

1. Spezieller Lieferschein als Begleitpapier (ADR-Daten)
2. Einfache Aus-/Einlagerung (Umlagerung)

Die Funktion liegt unter dem Menüpunkt „Lager/Umfuhren“.

Ablauf zur Erstellung von Umfuhr-Lieferscheinen

1. Empfangs- & Versandfirmierung auswählen.
2. Artikel einfügen.
   Folgende Besonderheiten sind zu beachten:
   1. Alle Artikel können ohne Einschränkung eingefügt werden (Artikel-Typ, Exklusiv).
   2. Menge muss manuell gesetzt werden (ungeachtet Verrechnungseinheit).
   3. Bei Artikeln ohne Gebinde muss dieses zusätzlich gewählt werden (für ADR-Daten).
3. Lagerausgang definieren.
   1. Zu jeder Position, in Spalte „Lagerausgang“, über ?-Icon ist die Menge zu einem Bestand zu wählen.
4. Umlagerung durchführen.
   1. Mit Klick auf „Umlagern“ muss das Ziellager ausgewählt werden Nach Auswahl werden die Lagerbewegungen durchgeführt.
   2. Der Umfuhr-Lieferschein wird als geschlossen markiert.

Hinweise

- Achten Sie bei der Verladung auf die richtigen Mengen und Chargen.
- Lagerausgang sollte kein Reservierungs- oder Produktionslager sein.
- Eigener Nummernkreis für Umfuhren sollte hinterlegt werden.
- Empfangsfirmierung wird durch einen Kunden in der Firmierung definiert.

Suche von Umfuhr-Lieferscheinen

Umfuhr-Lieferscheine werden unter „Lager/Umfuhren“ und „Verkauf/Lieferscheine“ angezeigt.

Vollständige Liste vorhandener BDEs

Jahresauswertung Angebot

Name: Jahresauswertung Angebot
Beschreibung: Jahresauswertung Angebote zu Auftrag

SQL-Abfrage (Abfrage nach Bearbeiter):

```
WITH angebote AS (
  SELECT
    TO_CHAR(oe.transdate, 'YYYY') AS jahr,
    COUNT(*) AS anzahl,
    SUM(oe.amount) AS summe
  FROM oe
  LEFT JOIN employee ON oe.employee_id = employee.id
  WHERE oe.customer_id IS NOT NULL
    AND COALESCE(oe.quotation, FALSE) = TRUE
    AND employee.id = <%Bearbeiter%>
  GROUP BY TO_CHAR(oe.transdate, 'YYYY')
),
auftraege AS (
  SELECT
    TO_CHAR(oe.transdate, 'YYYY') AS jahr,
    COUNT(*) AS anzahl,
    SUM(oe.amount) AS summe
  FROM oe
  LEFT JOIN employee ON oe.employee_id = employee.id
  WHERE oe.customer_id IS NOT NULL
    AND COALESCE(oe.quotation, FALSE) = FALSE
    AND employee.id = <%Bearbeiter%>
  GROUP BY TO_CHAR(oe.transdate, 'YYYY')
)
SELECT
  COALESCE(a.jahr, o.jahr) AS "Jahr",
```

```
  a.anzahl AS "Anzahl Angebote",
  a.summe  AS "Summe Angebote",
  ROUND(a.summe::numeric / NULLIF(a.anzahl, 0), 2) AS "Ø Angebote",

  o.anzahl AS "Anzahl Aufträge",
  o.summe  AS "Summe Aufträge",
  ROUND(o.summe::numeric / NULLIF(o.anzahl, 0), 2) AS "Ø Aufträge"

FROM angebote a
FULL JOIN auftraege o ON a.jahr = o.jahr
ORDER BY "Jahr";
```

SQL-Abfrage (Abfrage nach Verkäufer):

```
WITH angebote AS (
  SELECT
    TO_CHAR(oe.transdate, 'YYYY') AS jahr,
    COUNT(*) AS anzahl,
    SUM(oe.amount) AS summe
  FROM oe
  LEFT JOIN employee salesman ON oe.salesman_id = salesman.id
  WHERE oe.customer_id IS NOT NULL
    AND COALESCE(oe.quotation, FALSE) = TRUE
    AND salesman.id = <%Verkäufer%>
  GROUP BY TO_CHAR(oe.transdate, 'YYYY')
),
auftraege AS (
  SELECT
    TO_CHAR(oe.transdate, 'YYYY') AS jahr,
    COUNT(*) AS anzahl,
    SUM(oe.amount) AS summe
  FROM oe
  LEFT JOIN employee salesman ON oe.salesman_id = salesman.id
  WHERE oe.customer_id IS NOT NULL
    AND COALESCE(oe.quotation, FALSE) = FALSE
    AND salesman.id = <%Verkäufer%>
  GROUP BY TO_CHAR(oe.transdate, 'YYYY')
)
```

```
SELECT
  COALESCE(a.jahr, o.jahr) AS "Jahr",

  a.anzahl AS "Anzahl Angebote",
  a.summe  AS "Summe Angebote",
  ROUND(a.summe::numeric / NULLIF(a.anzahl, 0), 2) AS "Ø Angebote",

  o.anzahl AS "Anzahl Aufträge",
  o.summe  AS "Summe Aufträge",
  ROUND(o.summe::numeric / NULLIF(o.anzahl, 0), 2) AS "Ø Aufträge"

FROM angebote a
FULL JOIN auftraege o ON a.jahr = o.jahr
ORDER BY "Jahr";
```

Abfrageparameter:

- **Bearbeiter" bzw. "Verkäufer"**
  - **Typ:** "Bearbeiter"
  - **Beschreibung:** "Auswahl Bearbeiter" bzw. "Auswahl Verkäufer"
  - **Standardwert:** Kein Standardwert

Top Kunden im Zeitraum X

Name: Top Kunden im Zeitraum X

Beschreibung: Anzeige der umsatzstärksten Kunden (Kundenanzahl kann limitiert werden)

SQL-Abfrage:

```
SELECT
  customer.customernumber AS "Kundennummer",
  customer.name AS "Kundenname",
  total AS "Betrag"
FROM customer
INNER JOIN (
  SELECT sum(netamount) AS total, customer_id
  FROM ar
  WHERE NOT ar.storno
    AND ar.datepaid BETWEEN <%Anfangsdatum%> AND <%Endddatum%>
  GROUP BY customer_id
```

```
) ar ON ar.customer_id=customer.id
ORDER BY total DESC
LIMIT <%Limitierung%>
```

Abfrageparameter:

- **Anfangsdatum:**
  - **Typ:** Datum
  - **Beschreibung:** Beginn der gewünschten Zeitspanne
  - **Standardwert:** Kein Standardwert
- **Enddatum:**
  - **Typ:** Datum
  - **Beschreibung:** Ende der gewünschten Zeitspanne
  - **Standardwert:** Kein Standardwert
- **Limitierung:**
  - **Typ:** Nummer
  - **Beschreibung:** Limitierung der Anzahl der angezeigten Kunden
  - **Standardwert:** Kein Standardwert

Inaktive Kunden

Name: Inaktive Kunden

Beschreibung: Gültige Kunden, ohne Rechnung ab Datum
Ab einem gewählten Datum wird geprüft, ob es Rechnungen für einen Kunden gibt.
Falls nicht, werden diese aufgelistet, nach Nettobetrag sortiert.

SQL-Abfrage:

```
SELECT customernumber "Kundennummer", name AS "Kundenname", invnumber AS "Rechnungsnummer",
transdate AS "Rechnungsdatum", netamount AS "Nettobetrag"
FROM customer
LEFT JOIN (
  SELECT ar.customer_id, ar.invnumber, ar.transdate, ar.netamount
  FROM ar
  INNER JOIN (
    SELECT max(id) AS maxid, customer_id
      FROM ar AS ar1
      WHERE transdate = (SELECT max(transdate) AS maxtrans FROM ar AS ar2 WHERE
ar1.customer_id = ar2.customer_id)
    GROUP BY customer_id) maxAR
  ON ar.customer_id = maxAR.customer_id
```

```
  WHERE ar.id = maxAR.maxid
    AND ar.customer_id NOT IN (
      SELECT distinct customer_id FROM ar WHERE transdate >= <%Zeitpunkt%>)) AS max_inv
ON max_inv.customer_id = customer.id
WHERE max_inv.invnumber IS NOT null
ORDER BY netamount DESC
```

Abfrageparameter:

- **Zeitpunkt:**
  - **Typ:** Datum
  - **Beschreibung:** Zeitpunkt ab dem der Kunde auf Inaktivität geprüft wird
  - **Standardwert:** Kein Standardwert

Aktive Kunden mit zugehöriger Lieferbedingung

Name: Aktive Kunden mit zugehöriger Lieferbedingung

Beschreibung: -

SQL-Abfrage:

```
SELECT
  customer.customernumber AS "Kundennummer",
  customer.name AS "Kundenname",
  dt.description AS "Lieferbedingung"
--  total AS "Betrag"
FROM customer
INNER JOIN (
  SELECT sum(netamount) AS total, customer_id
  FROM ar
  WHERE NOT ar.storno
    AND ar.datepaid BETWEEN <%Anfangsdatum%> AND <%Endddatum%>
  GROUP BY customer_id
) ar ON ar.customer_id=customer.id
JOIN delivery_terms dt ON customer.delivery_term_id = dt.id
-- ORDER BY total DESC;
```

Abfrageparameter:

- **Anfangsdatum:**

    - **Typ:** Datum
    - **Beschreibung:** -
    - **Standardwert:** Kein Standardwert
- **Enddatum:**
    - **Typ:** Datum
    - **Beschreibung:** -
    - **Standardwert:** Kein Standardwert

Kundenumsätze nach Vertreter

Name: Kundenumsätze nach Vertreter

Beschreibung: Ausgabe von Umsätzen (enthalten auch nicht-provisionierbare Umsätze), Einkaufswerten und Marge je Kunde für einen bestimmten Zeitraum und Vertreter

SQL-Abfrage:

```
SELECT customer.customernumber as Kundennummer, customer.name as Name, ROUND(sales,2) as
Umsatz, ROUND(sales-margin,2) as Einkaufswert, ROUND(margin,2) as Marge, CASE WHEN sales=0
THEN 0 ELSE ROUND(((margin/sales)*100),2) END as prozentual
FROM (
  SELECT invoice.customer_id customer_id, SUM(netamount) sales, SUM(marge_total) margin
  FROM (
    SELECT *
    FROM ar
    WHERE transdate BETWEEN <%Anfang%> AND <%Ende%>
  ) invoice
  JOIN (
    SELECT customer_id, startdate, LEAD(startdate, 1) OVER (PARTITION BY customer_id ORDER BY
startdate) AS enddate
    FROM provisioning_rules
    WHERE parts_id is null and partsgroup_id is null and vendor_id=(SELECT id FROM vendor
WHERE id=<%Vertreter%>)
  ) customer_periods
  ON invoice.customer_id=customer_periods.customer_id AND
    invoice.transdate>=customer_periods.startdate AND
    (customer_periods.enddate IS NULL OR invoice.transdate<customer_periods.enddate)
  GROUP BY invoice.customer_id
) customer_margin
JOIN customer ON customer_margin.customer_id=customer.id
```

```
ORDER BY Umsatz DESC;
```

Abfrageparameter:

- **Anfang:**
  - **Typ:** Datum
  - **Beschreibung:** Anfangsdatum des Zeitraums
  - **Standardwert:** Kein Standardwert
- **Ende:**
  - **Typ:** Datum
  - **Beschreibung:** Enddatum des Zeitraums
  - **Standardwert:** Kein Standardwert
- **Vertreter:**
  - **Typ:** Nummer
  - **Beschreibung:** Vertreter
  - **Standardwert:** Kein Standardwert

Kundenumsätze nach Vertreter mit Vorjahresvergleich

Name: Kundenumsätze nach Vertreter mit Vorjahresvergleich

Beschreibung: Ausgabe von Umsätzen (enthalten auch nicht-provisionierbare Umsätze) eines Vertreters pro Kunde mit Vergleich zum gleichen Zeitraum im Vorjahr

SQL-Abfrage:

```
WITH customer_periods(customer_id, startdate, enddate) AS (
  SELECT customer_id, startdate, LEAD(startdate, 1) OVER (PARTITION BY customer_id ORDER BY
startdate) AS enddate
  FROM provisioning_rules
  WHERE parts_id is null and partsgroup_id is null and vendor_id=(SELECT id FROM vendor WHERE
id=<%Vertreter%>)
)

SELECT
  (SELECT vendornumber FROM vendor WHERE id=<%Vertreter%>) AS "Vertreternummer",
  (SELECT name FROM vendor WHERE id=<%Vertreter%>) AS "Vertretername",
  customer.customernumber as "Kundennummer",
  customer.name as "Kundenname",
  ROUND(customer_margin.sales,2) as "Umsatz aktueller Zeitraum",
  ROUND(customer_margin_before.sales,2) as "Umsatz Vorjahr",
  CASE WHEN customer_margin_before.sales=0 THEN 0 ELSE
```

```
customer_margin.sales*100/customer_margin_before.sales END AS "Umsatzveränderung in %"
FROM (
  SELECT invoice.customer_id customer_id, SUM(netamount) sales, SUM(marge_total) margin
  FROM (
    SELECT *
    FROM ar
    WHERE transdate BETWEEN <%Anfang%> AND <%Ende%>
  ) invoice
  JOIN customer_periods
  ON invoice.customer_id=customer_periods.customer_id AND
    invoice.transdate>=customer_periods.startdate AND
    (customer_periods.enddate IS NULL OR invoice.transdate<customer_periods.enddate)
  GROUP BY invoice.customer_id
) customer_margin
FULL JOIN (
  SELECT invoice.customer_id customer_id, SUM(netamount) sales, SUM(marge_total) margin
  FROM (
    SELECT *
    FROM ar
    WHERE transdate BETWEEN (<%Anfang%>::date - interval '1 year') AND (<%Ende%>::date -
interval '1 year')
  ) invoice
  JOIN customer_periods
  ON invoice.customer_id=customer_periods.customer_id AND
    invoice.transdate>=customer_periods.startdate AND
    (customer_periods.enddate IS NULL OR invoice.transdate<customer_periods.enddate)
  GROUP BY invoice.customer_id
) customer_margin_before ON customer_margin.customer_id=customer_margin_before.customer_id
JOIN customer ON customer_margin.customer_id=customer.id OR
customer_margin_before.customer_id=customer.id
ORDER BY "Umsatz aktueller Zeitraum" DESC;
```

Abfrageparameter:

- **Anfang:**
  - **Typ:** Datum
  - **Beschreibung:** -
  - **Standardwert:** Kein Standardwert
- **Ende:**
  - **Typ:** Datum

- **Beschreibung:** -
- **Standardwert:** Kein Standardwert

- **Vertreter:**
  - **Typ:** Nummer
  - **Beschreibung:** -
  - **Standardwert:** Kein Standardwert

Kundenumsätze nach Vertreter mit Vorjahresvergleich und Deckungsbetrag prozentual

Name: Kundenumsätze nach Vertreter mit Vorjahresvergleich und Deckungsbetrag

Beschreibung: Ausgabe von Umsätzen (enthalten auch nicht-provisionierbare Umsätze) eines Vertreters pro Kunde mit Vergleich zum gleichen Zeitraum im Vorjahr

SQL-Abfrage:

```
WITH customer_periods(customer_id, startdate, enddate) AS (
  SELECT customer_id, startdate, LEAD(startdate, 1) OVER (PARTITION BY customer_id ORDER BY
startdate) AS enddate
  FROM provisioning_rules
  WHERE parts_id is null and partsgroup_id is null and vendor_id=(SELECT id FROM vendor WHERE
id=<%Vertreter%>)
)

SELECT
  (SELECT vendornumber FROM vendor WHERE id=<%Vertreter%>) AS "Vertreternummer",
  (SELECT name FROM vendor WHERE id=<%Vertreter%>) AS "Vertretername",
  customer.customernumber as "Kundennummer",
  customer.name as "Kundenname",
  ROUND(customer_margin.sales,2) as "Umsatz aktueller Zeitraum",
  ROUND(customer_margin_before.sales,2) as "Umsatz Vorjahr",
  CASE WHEN customer_margin_before.sales=0 THEN 0 ELSE
customer_margin.sales*100/customer_margin_before.sales END AS "Umsatzveränderung in %",
  CASE WHEN customer_margin.sales=0        THEN 0 ELSE
ROUND(customer_margin.margin*100/customer_margin.sales,2) END AS "Deckungsbeitrag aktueller
Zeitraum",
  CASE WHEN customer_margin_before.sales=0 THEN 0 ELSE
ROUND(customer_margin_before.margin*100/customer_margin_before.sales,2) END AS
"Deckungsbeitrag Vorjahr"
```

```
FROM (
  SELECT
    invoice.customer_id customer_id,
    SUM(netamount) sales,
    SUM(marge_total) margin
  FROM (
    SELECT *
    FROM ar
    WHERE transdate BETWEEN <%Anfang%> AND <%Ende%>
  ) invoice
  JOIN customer_periods
  ON invoice.customer_id=customer_periods.customer_id AND
    invoice.transdate>=customer_periods.startdate AND
    (customer_periods.enddate IS NULL OR invoice.transdate<customer_periods.enddate)
  GROUP BY invoice.customer_id
) customer_margin
FULL JOIN (
  SELECT
    invoice.customer_id customer_id,
    SUM(netamount) sales,
    SUM(marge_total) margin
  FROM (
    SELECT *
    FROM ar
    WHERE transdate BETWEEN (<%Anfang%>::date - interval '1 year') AND (<%Ende%>::date -
interval '1 year')
  ) invoice
  JOIN customer_periods
  ON invoice.customer_id=customer_periods.customer_id AND
    invoice.transdate>=customer_periods.startdate AND
    (customer_periods.enddate IS NULL OR invoice.transdate<customer_periods.enddate)
  GROUP BY invoice.customer_id
) customer_margin_before ON customer_margin.customer_id=customer_margin_before.customer_id
JOIN customer ON customer_margin.customer_id=customer.id OR
customer_margin_before.customer_id=customer.id
ORDER BY "Umsatz aktueller Zeitraum" DESC;
```

Abfrageparameter:

- **Anfang:**
  - **Typ:** Datum
  - **Beschreibung:** -
  - **Standardwert:** Kein Standardwert
- **Ende:**
  - **Typ:** Datum
  - **Beschreibung:** -
  - **Standardwert:** Kein Standardwert
- **Vertreter:**
  - **Typ:** Nummer
  - **Beschreibung:** -
  - **Standardwert:** Kein Standardwert

Anzeige aller Angebote eines Kunden

Name: Anzeige aller Angebote eines Kunden

Beschreibung: Anzeige aller Angebote, zu einem Kunden, mit Auflistung:
Gesamtanzahl Angebote / noch offen / Angenommene Angebote / Abgelehnte Angebote

SQL-Abfrage:

```
SELECT alle.count as "Gesamtanzahl Angebote", offen.count as "noch offen", angenommen.count as
"Angenommene Angebote", abgelehnt.count as "Abgelehnte Angebote"
FROM
(SELECT count(id) count FROM oe where customer_id = <%Kunde%> AND quotation is true ) alle,
(SELECT count(id) count FROM oe where customer_id = <%Kunde%> AND quotation is true AND closed
= false ) offen,
(SELECT count(id) count FROM oe where customer_id = <%Kunde%> AND quotation is true AND closed
= true AND id in (SELECT from_id FROM record_links WHERE from_table = 'oe' AND to_table = 'oe'
AND to_id IN (SELECT id from oe WHERE customer_id IS NOT NULL and quotation is false)))
angenommen,
(SELECT count(id) count FROM oe where customer_id = <%Kunde%> AND quotation is true AND closed
= true AND id not in (SELECT from_id FROM record_links WHERE from_table = 'oe' AND to_table =
'oe'
AND to_id IN (SELECT id from oe WHERE customer_id IS NOT NULL and quotation is false)))
abgelehnt;
```

Abfrageparameter:

- **Kunde:**
  - **Typ:** Text

- **Beschreibung:** Kunde
- **Standardwert:** Kein Standardwert

Verkaufte Artikel in Zeitraum für einen Kunden

Name: Verkaufte Artikel in Zeitraum für einen Kunden

Beschreibung: Auswertung aller verkauften Artikel für einen Kunden mit Ausgabe von Rechnungsnummer, Datum,
Verkaufsbetrag, Einkaufsbetrag und Ertrag in einem bestimmten Zeitraum.

SQL-Abfrage:

```
SELECT partnumber AS "Artikelnummer",description AS "Artikelbeschreibung", invnumber AS
"Re.Nr",
transdate AS "Re.Datum", vkbetrag AS "VK-Betrag", ekbetrag AS "EK-Betrag", vkbetrag -
ekbetrag  AS "Ertrag", text_value  AS "Hersteller"
FROM (
SELECT
    p.partnumber,
    i.description,
    a.invnumber,
    a.transdate,
    c.name,
ROUND(i.qty * i.fxsellprice * (1 - i.discount::numeric) / COALESCE(i.price_factor, 1), 2) AS
vkbetrag,
ROUND(i.qty * i.lastcost, 2) AS ekbetrag,
    cvar.text_value,
    p.id,
    i.parts_id,
    i.trans_id,
    a.id,
    c.id
FROM "parts" AS p
INNER JOIN "invoice" AS i
ON p.id = i.parts_id
INNER JOIN "ar" a
ON i.trans_id = a.id
INNER JOIN "customer" c
ON a.customer_id = c.id
```

```
LEFT JOIN "custom_variables" cvar
ON p.id = cvar.trans_id AND cvar.config_id = 5
  AND cvar.sub_module = ''
WHERE
  c.id = <%Kunde%>
  AND a.transdate >= <%Anfangsdatum%>
  AND a.transdate <= <%Enddatum%>
ORDER BY a.invnumber) AS t;
```

Abfrageparameter:

- **Anfangsdatum:**
  - **Typ:** Datum
  - **Beschreibung:** Anfangsdatum
  - **Standardwert:** Kein Standardwert
- **Enddatum:**
  - **Typ:** Datum
  - **Beschreibung:** Enddatum
  - **Standardwert:** Kein Standardwert
- **Kunde:**
  - **Typ:** Text
  - **Beschreibung:** -
  - **Standardwert:** Kein Standardwert

Deckungsbeitrag eines Zeitraums für einen bestimmten Kunden

Name: Deckungsbeitrag eines Zeitraums für einen bestimmten Kunden

Beschreibung: Ausgabe von Artikel mit VK- und EK-Preis und deren Deckungsbeitrag unter Beachtung des Rabattes aus den Rechnungen mit: Rechnungsnummer, Position, Artikelnummer, Artikelbeschreibung, VK-Preis, EK-Preis, Rabatt (%), Menge, Deckungsbeitrag (%)

SQL-Abfrage:

```
SELECT
  ar.invnumber AS "Rechnungsnummer",
  invoice.position AS "Position",
  parts.partnumber AS "Artikelnummer",
  invoice.description AS "Artikelbeschreibung",
  invoice.sellprice AS "VK-Preis",
```

```
  parts.lastcost "EK-Preis",
  invoice.discount::numeric * 100 AS "Rabatt (%)",
  invoice.qty AS "Menge",
  CASE WHEN invoice.sellprice=0
    THEN 0
    ELSE ROUND(100*(invoice.sellprice * (1 - invoice.discount::numeric) -
parts.lastcost)/invoice.sellprice, 2)
  END AS "Deckungsbeitrag (%)"
FROM invoice
JOIN parts ON invoice.parts_id=parts.id
JOIN ar ON invoice.trans_id=ar.id
WHERE ar.transdate BETWEEN <%Anfang%> AND <%Ende%> AND ar.customer_id=<%Kunde%>
ORDER BY partnumber
```

Abfrageparameter:

- **Anfang:**
  - **Typ:** Datum
  - **Beschreibung:** von
  - **Standardwert:** Kein Standardwert
- **Ende:**
  - **Typ:** Datum
  - **Beschreibung:** bis
  - **Standardwert:** Kein Standardwert
- **Kunde:**
  - **Typ:** Nummer
  - **Beschreibung:** Kunde
  - **Standardwert:** Kein Standardwert

Lieferanten mit Steuerzone "Inland" und Länderkürzel ungleich "DE"

Name: Lieferanten mit Steuerzone "Inland" und Länderkürzel ungleich "DE"

Beschreibung: Alle Lieferanten mit Steuerzone "Inland" deren Länderkürzel nicht "DE" entspricht ausgegeben mit:
Lieferantennummer, Lieferantenname, Steuerzone, Länderkürzel

SQL-Abfrage:

```
SELECT v.vendornumber AS "Lieferantennummer",
    v.name AS "Lieferantenname",
    t.description AS "Steuerzone",
    c.iso AS "Länderkürzel"
FROM vendor v,
    countries c,
    tax_zones t
WHERE v.taxzone_id = 4
    AND v.country_id != (
        SELECT id
        FROM countries
        WHERE iso = 'DE'
    )
    AND v.taxzone_id = t.id
    AND v.country_id = c.id;
```

Lieferanten mit Steuerzone ungleich "Inland" und Länderkürzel "DE"

Name: Lieferanten mit Steuerzone ungleich "Inland" und Länderkürzel "DE"

Beschreibung: Lieferanten mit Steuerzone ungleich "Inland" und Länderkürzel "DE" mit: Lieferantennummer, Lieferantenname, Steuerzone, Länderkürzel

SQL-Abfrage:

```
SELECT v.vendornumber AS "Lieferantennummer",
    v.name AS "Lieferantenname",
    t.description AS "Steuerzone",
    c.iso AS "Länderkürzel"
FROM vendor v,
    countries c,
    tax_zones t
WHERE v.taxzone_id != 4
    AND v.country_id = (
        SELECT id
        FROM countries
        WHERE iso = 'DE'
    )
```

```
    AND v.taxzone_id = t.id
    AND v.country_id = c.id;
```

Preisregelexport - Kundenpreise

Name: Preisregelexport - Kundenpreise

Beschreibung: Ausgabe der Kundenpreise, welche zum Zeitpunkt des Exports greifen

SQL-Abfrage:

```
SELECT
  COALESCE(NULLIF(pc.customer_partnumber,''),p.partnumber) AS "Artikelnummer",
  COALESCE(NULLIF(pc.customer_partdescription,''),p.description) AS "Artikelbeschreibung",
  CASE WHEN lower(pr.qty) IS NULL
       THEN ''
       ELSE CASE WHEN lower_inc(pr.qty) THEN '>=' ELSE '>' END || ' ' || lower(pr.qty)
  END ||
  CASE WHEN lower(pr.qty) IS NULL OR upper(pr.qty) IS NULL THEN '' ELSE ' ' END ||
  CASE WHEN upper(pr.qty) IS NULL
       THEN ''
       ELSE CASE WHEN upper_inc(pr.qty) THEN '<=' ELSE '<' END || ' ' || upper(pr.qty)
  END AS "Menge",
  pr.price AS "Preis",
  pr.reduction AS "Abschlag",
  pr.discount AS "Rabatt"
FROM parts p
JOIN part_customer_prices pc ON pc.parts_id=p.id
JOIN price_rules pr ON pr.part_id=p.id AND pr.customer_id=pc.customer_id
WHERE pc.customer_id=<%Kunde%> AND pr.obsolete=FALSE AND pr.type='customer'
  AND (pr.transdate IS NULL OR pr.transdate @> current_date);
```

Abfrageparameter:

- **Kunde:**
  - **Typ:** Nummer
  - **Beschreibung:** Kunde für Kundenpreise
  - **Standardwert:** Kein Standardwert

Preisregelexport - Spezifische Preisregel

Name: Preisregelexport - Spezifische Preisregel

Beschreibung: Ausgabe der Kundenpreise einer bestimmten Preisregel, welche zum Zeitpunkt des Exports greifen mit:
Kundennummer, Kundenname, Artikelnummer, Artikelbeschreibung, Menge, Preis, Abschlag, Rabatt

SQL-Abfrage:

```
SELECT
  COALESCE (c.customernumber) AS "Kundennummer",
  COALESCE (c.name) AS "Kundenname",
  COALESCE(NULLIF(pc.customer_partnumber,''),p.partnumber) AS "Artikelnummer",
  COALESCE(NULLIF(pc.customer_partdescription,''),p.description) AS "Artikelbeschreibung",
  CASE WHEN lower(pr.qty) IS NULL
       THEN ''
       ELSE CASE WHEN lower_inc(pr.qty) THEN '>=' ELSE '>' END || ' ' || lower(pr.qty)
  END ||
  CASE WHEN lower(pr.qty) IS NULL OR upper(pr.qty) IS NULL THEN '' ELSE ' ' END ||
  CASE WHEN upper(pr.qty) IS NULL
       THEN ''
       ELSE CASE WHEN upper_inc(pr.qty) THEN '<=' ELSE '<' END || ' ' || upper(pr.qty)
  END AS "Menge",
  pr.price AS "Preis",
  pr.reduction AS "Abschlag",
  pr.discount AS "Rabatt"
FROM parts p
JOIN part_customer_prices pc ON pc.parts_id=p.id
JOIN customer c ON c.id=pc.customer_id
JOIN price_rules pr ON pr.part_id=p.id AND pr.customer_id=pc.customer_id
WHERE pr.name ILIKE <%Preisregelname%> AND pr.obsolete=FALSE AND pr.type='customer'
  AND (pr.transdate IS NULL OR pr.transdate @> current_date);
```

Abfrageparameter:

- **Preisregelname:**
  - **Typ:** Text
  - **Beschreibung:** Zur Suche nach mehreren Preisregeln kann % als Platzhalter genutzt werden
  - **Standardwert:** Kein Standardwert

Kunden und Vertreterdaten zu Rechnungen aus bestimmten Zeitraum

Name: Kunden und Vertreterdaten zu Rechnungen aus bestimmten Zeitraum

Beschreibung: Ausgabe der folgenden Daten: Vertreternr., Vertretername, Kundennummer, Kundenname, Straße, PLZ, Ort, Land

SQL-Abfrage:

```
SELECT
  v.vendornumber AS "Vertreternr.",
  v.name AS "Vertretername",
  c.customernumber AS "Kundennummer",
  c.name AS "Kundenname",
  c.street AS "Straße",
  c.zipcode AS "PLZ",
  c.city AS "Ort",
  c.country AS "Land"
FROM
  customer c
  LEFT OUTER JOIN provisioning_rules p ON c.id = p.customer_id
  LEFT OUTER JOIN vendor v ON (p.vendor_id = v.id)
  JOIN ar ON ar.customer_id = c.id
WHERE
  ar.transdate > <%Anfangsdatum%>
  AND ar.transdate <= <%Enddatum%>
GROUP by
  c.customernumber,
  v.vendornumber,
  v.name,
  c.name,
  c.street,
  c.zipcode,
  c.city,
  c.country
ORDER BY c.customernumber;
```

Abfrageparameter:

- **Anfangsdatum:**

  - **Typ:** Datum
  - **Beschreibung:** -
  - **Standardwert:** Kein Standardwert
- **Enddatum:**
  - **Typ:** Datum
  - **Beschreibung:** -
  - **Standardwert:** Kein Standardwert

Weihnachtsliste - Vorprüfung

> Dieser BDE benötigt die benutzerdefinierte Variable "cp_calender" für Ansprechpersonen

Name: Weihnachtsliste - Vorprüfung

Beschreibung: Vor der Erstellung der Weihnachtsliste sind folgende Informationen wichtig:
1) Gibt es Kunden ohne Ansprechperson?
2) Gibt es Kunden, wo keine Ansprechperson die Variable "Weihnachtskalender" gesetzt hat?
Beide Fälle werden von dem Bericht "Weihnachtsliste - Erstellung" nicht beachtet.

Damit diese Kunden nicht übersehen werden, erfolgt dieser Bericht:

SQL-Abfrage:

```
--Keine Ansprechperson oder keine Ansprechperson mit CheckBox
SELECT
  regexp_replace(cus.customernumber, '\[\[.*\]\]', '') AS Kundennummer,
  regexp_replace(cus.name, '\[\[.*\]\]', '') AS Kundenname,
  regexp_replace(cus.street, '\[\[.*\]\]', '') AS Straße,
  regexp_replace(cus.zipcode, '\[\[.*\]\]', '') AS PLZ,
  regexp_replace(cus.city, '\[\[.*\]\]', '') AS Stadt,
  (SELECT max(transdate) FROM ar WHERE customer_id = cus.id AND NOT storno) AS letzte_Rechnung
FROM
  customer cus
LEFT JOIN contacts con
  ON cus.id = con.cp_cv_id
LEFT JOIN custom_variables cv
  ON con.cp_id = cv.trans_id
  AND cv.config_id = 23 --cp_calender
  AND cv.bool_value = TRUE
GROUP BY cus.id, cus.customernumber, cus.name, cus.street, cus.zipcode, cus.city, con.cp_cv_id
```

```
HAVING
  (COUNT(cv.trans_id) = 0 OR con.cp_cv_id IS NULL)
ORDER BY cus.name;
```

Weihnachtsliste - Erstellung

> Dieser BDE benötigt die benutzerdefinierte Variable "cp_calender" für Ansprechpersonen.

Name: Weihnachtsliste - Erstellung

Beschreibung: Auflistung von Ansprechpersonen mit gesetzter Variable "Weihnachtskalender". Folgende Spalten werden ausgegeben:
Kunden-Nr. -Name, Anrede, Vorname, Nachname, Str. PLZ, Stadt, Tel, Email, Datum von der letzten Rechnung.

SQL-Abfrage:

```
SELECT
  regexp_replace(cus.customernumber, '\[\[.*\]\]', '') AS Kundennummer,
  regexp_replace(cus.name, '\[\[.*\]\]', '') AS Kundenname,
  CASE
    WHEN con.cp_gender = 'f' THEN 'Frau'
    WHEN con.cp_gender = 'm' THEN 'Herr'
  END AS Anrede,
  regexp_replace(con.cp_givenname, '\[\[.*\]\]', '') AS Vorname,
  regexp_replace(con.cp_name, '\[\[.*\]\]', '') AS Nachname,
  COALESCE(NULLIF(regexp_replace(con.cp_street, '\[\[.*\]\]', ''), ''),
regexp_replace(cus.street, '\[\[.*\]\]', '')) AS Straße,
  COALESCE(NULLIF(regexp_replace(con.cp_zipcode, '\[\[.*\]\]', ''), ''),
regexp_replace(cus.zipcode, '\[\[.*\]\]', '')) AS PLZ,
  COALESCE(NULLIF(regexp_replace(con.cp_city, '\[\[.*\]\]', ''), ''), regexp_replace(cus.city,
'\[\[.*\]\]', '')) AS Stadt,
  regexp_replace(con.cp_phone1, '\[\[.*\]\]', '') AS Tel,
  regexp_replace(con.cp_email, '\[\[.*\]\]', '') AS EMAIL,
  (SELECT max(transdate) FROM ar WHERE customer_id = cus.id AND NOT storno) AS letzte_Rechnung
FROM
  contacts con
  JOIN customer cus ON (id = con.cp_cv_id)
WHERE
  con.cp_cv_id IN (
```

```
    SELECT
      id
    FROM
      customer
  )
  AND con.cp_id IN (
    SELECT
      trans_id
    FROM
      custom_variables
    WHERE
      config_id = 23
      AND bool_value = TRUE
  )
ORDER BY
  cus.name;
```