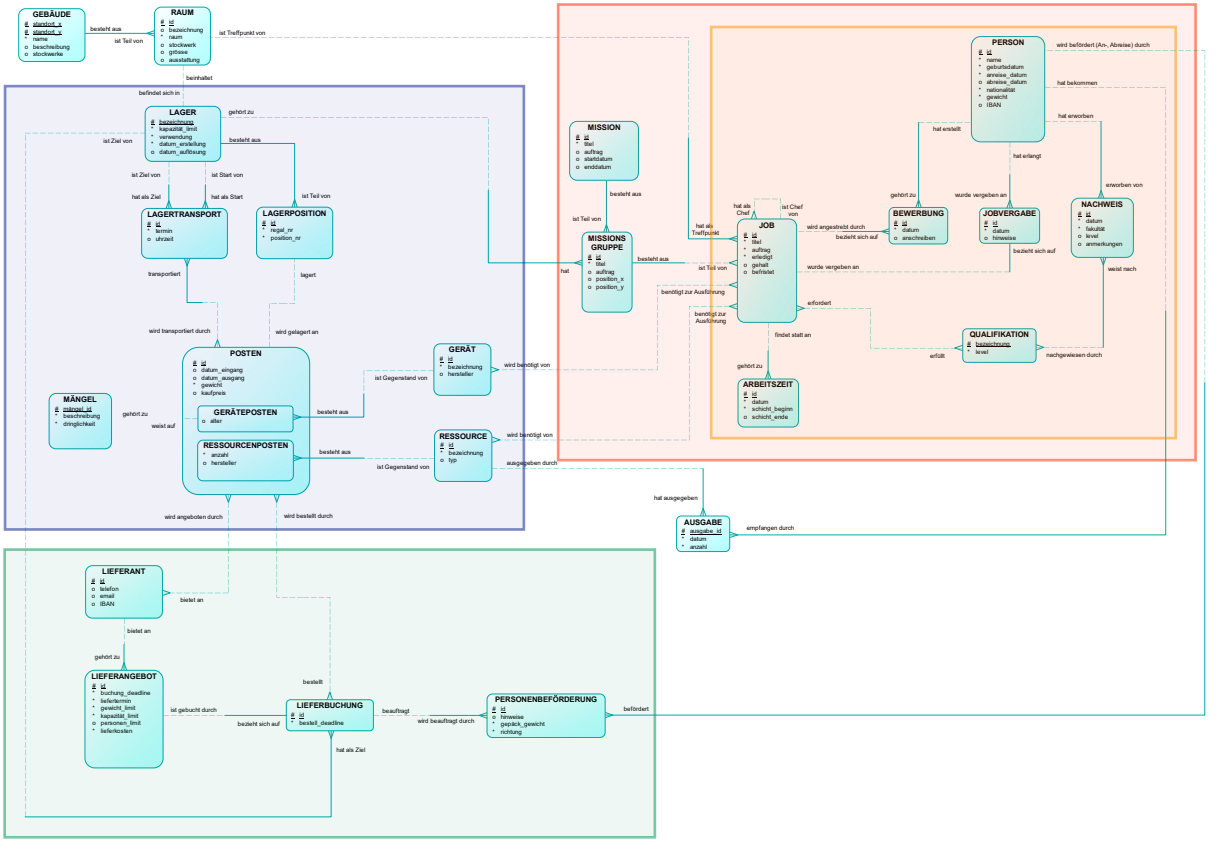
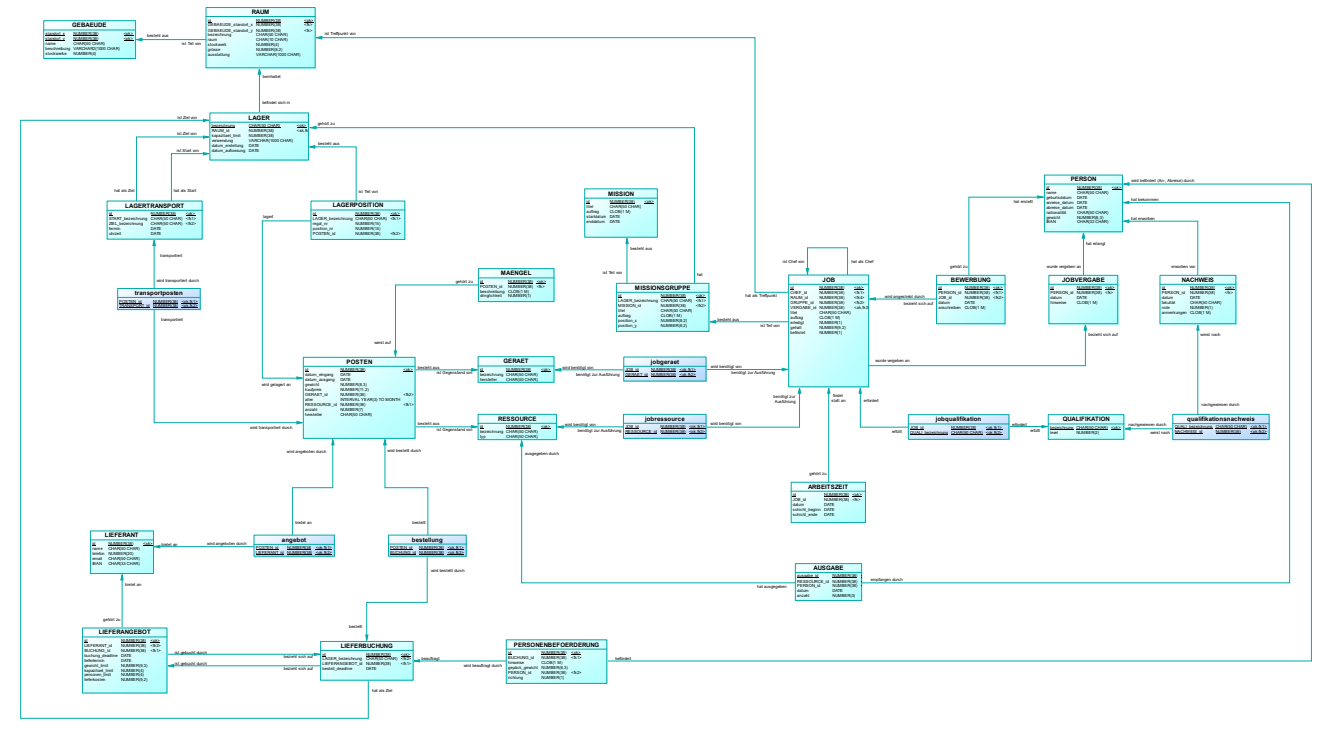

GEBÄUDE
standort_x
standort_y
name
beschreibung
stockwerke
besteht aus
ist Teil von
RAUM
bezeichnung
raum
stockwerk
größe
ausstattung
ist Treffpunkt von
beinhaltet
befindet sich in
LAGER
bezeichnung
kapazität_limit
verwendung
datum_erstellung
datum_auflösung
gehört zu
besteht aus
ist Ziel von
ist Start von
hat als Ziel
hat als Start
ist Teil von
LAGERTRANSPORT
termin
uhrzeit
LAGERPOSITION
regal_nr
position_nr
transportiert
lagert
wird transportiert durch
wird gelagert an
POSTEN
datum_eingang
datum_ausgang
gewicht
kaufpreis
GERÄTEPOSTEN
alter
RESSOURCENPOSTEN
anzahl
hersteller
besteht aus
ist Gegenstand von
GERÄT
bezeichnung
hersteller
wird benötigt von
RESSOURCE
bezeichnung
typ
wird benötigt von
ausgegeben durch
MÄNGEL
mängel_id
beschreibung
dringlichkeit
gehört zu
weist auf
wird angeboten durch
wird bestellt durch
MISSION
titel
auftrag
startdatum
enddatum
besteht aus
ist Teil von
MISSIONS GRUPPE
titel
auftrag
position_x
position_y
hat
besteht aus
ist Teil von
JOB
titel
auftrag
erledigt
gehalt
befristet
hat als Treffpunkt
hat als Chef
ist Chef von
benötigt zur Ausführung
benötigt zur Ausführung
wird angestrebt durch
bezieht sich auf
wurde vergeben an
wurde vergeben an
erfordert
findet statt an
PERSON
name
geburtsdatum
anreise_datum
abreise_datum
gewicht
nationalität
IBAN
wird befördert (An-, Abreise) durch
hat bekommen
hat erworben
hat erstellt
hat erlangt
BEWERBUNG
datum
anschreiben
gehört zu
JOBVERGABE
datum
hinweise
wurde vergeben an
bezieht sich auf
NACHWEIS
datum
fakultät
level
anmerkungen
erworben von
weist nach
QUALIFIKATION
bezeichnung
level
erfüllt
nachgewiesen durch
ARBEITSZEIT
datum
schicht_beginn
schicht_ende
gehört zu
hat ausgegeben
AUSGABE
ausgabe_id
datum
anzahl
empfangen durch
LIEFERANT
lieferant_id
telefon
email
IBAN
bietet an
bietet an
gehört zu
LIEFERANGEBOT
buchung_deadline
liefertermin
gewicht_limit
kapazität_limit
personen_limit
lieferkosten
ist gebucht durch
bezieht sich auf
bestellt
LIEFERBUCHUNG
bestell_deadline
beauftragt
wird beauftragt durch
PERSONENBEFÖRDERUNG
hinweise
gepäck_gewicht
richtung
befördert
hat als Ziel


GEBAEUDE
RAUM
LAGER
LAGERTRANSPORT
LAGERPOSITION
transportfirma
MISSION
MISSIONSGRUPPE
MAENGEL
PERSON
BEWERBUNG
JOBVERGABE
NACHWEIS
JOB
POSTEN
GERAET
jobgeraet
RESSOURCE
jobressource
jobqualifikation
QUALIFIKATION
qualifikationsnachweis
ARBEITSZEIT
LIEFERANT
angebot
bestellung
AUSGABE
LIEFERANGEBOT
LIEFERBUCHUNG
PERSONENBEFOERDERUNG

Partitionsschema und Indexierung

Beispiele für partionierte und indexierte Tabellen

AUSGABE

```
/* Table: AUSGABE                                              */
/*==============================================================*/
create table AUSGABE (
   ID                   NUMBER(38)            not null,
   RESSOURCE_ID         NUMBER(38)            not null,
   PERSON_ID            NUMBER(38)            not null,
   DATUM                DATE                 default SYSDATE  not null,
   ANZAHL               NUMBER(3)            default 1  not null
)
   partition by range(DATUM) interval(numtomyinterval(1, MONTH))
   subpartition by hash(RESSOURCE_ID) (
       partition AUSGABE_2024
       values less than (to_date('2025-01-01', 'YYYY-MM-DD'))
   )
/

comment on table AUSGABE is
```

Partitionen für jeden Monat (*partition by range*) automatisch erzeugt (*interval*), Subpartition nach dem Hash-Wert der ausgebenen *RESSOURCE*, somit kann bei Zugriffen auf *AUSGABEN* bestimmter *RESSOURCEN* in bestimmtem Zeitraum von Partition-Pruning profitiert werden

```
/* Index: IDX_AUSGABE_RESS_DATUM_LOC                           */
/*==============================================================*/
create index IDX_AUSGABE_RESS_DATUM_LOC on AUSGABE (
   RESSOURCE_ID ASC,
   DATUM ASC
)
local
/

/*==============================================================*/
/* Index: IDX_AUSGABE_DATUM_GLO                                 */
/*==============================================================*/
create index IDX_AUSGABE_DATUM_GLO on AUSGABE (
   DATUM ASC
)
global
```

lokaler Index auf die ausgegebenen *RESSOURCE* (primär) und das Datum (sekundär), somit kann effizient nach Ausgaben einer Ressource zu bestimmten Zeiträumen / Zeitpunkten innerhalb eines Monats gesucht werden

globaler Index auf das Datum, somit kann besser nach *AUSGABEN* bestimmter Zeiträume / Zeitpunkte unabhängig der *RESSOURCE* gesucht und sortiert werden

POSTEN

```
   end) virtual
)
   partition by list
    (TYP)
       (
           partition POSTEN_ANGEBOT
    values ('ANGEBOT'),
           partition POSTEN_LAGER_BESTAND
    values ('LAGER_BESTAND'),
           partition POSTEN_LAGER_ALT
    values ('LAGER_ALT')
       )
/

comment on table POSTEN is
'ein Posten im Lager (im Bestand / alt) oder ein Angebot eines Lieferanten,
```

Partitionen für die 3 Arten von POSTEN (Posten im Lager, ehemalige Lagerposten, Produkt-Angebote von Lieferanten) - da immer nur eine der Kategorien bei einer Abfrage gefragt ist, kann hier von Partition-Pruning profitert werden, außerdem könnten in einer Real-anwendung die Datenbankzugriffe der einzelnen Bereiche (Lagerverwaltung / Planung von Bestellungen) besser parallelisiert werden

```
/*==============================================================*/
/* Index: IDX_POSTEN_ID_LOC                                     */
/*==============================================================*/
create unique index IDX_POSTEN_ID_LOC on POSTEN (
   ID ASC
)
local
/

/*==============================================================*/
/* Index: IDX_POSTEN_PRODUCT_ID                                 */
/*==============================================================*/
create index IDX_POSTEN_PRODUCT_ID on POSTEN (
   COALESCE(RESSOURCE_ID, GERAET_ID) ASC,
   ID ASC
)
local
```

lokaler Index auf die ID, neben dem globalen Primärindex, somit kann innerhalb einer Kategorie nach einer ID gesucht werden, nur 1/3 des Umfangs des globalen Index wird durchsucht

lokaler Index auf die Produkt ID (entweder *RESSOURCE_id* der *GERAET_id*), somit kann gezielt nach Produkten gesucht werden

Partitionsschema und Indexierung

Beispiele für partionierte und indexierte Tabellen

LAGERPOSITION

```
drop table LAGERPOSITION cascade constraints
/

/*==============================================================*/
/* Table: LAGERPOSITION                                         */
/*==============================================================*/
create table LAGERPOSITION (
   ID                   NUMBER(38)            not null,
   LAGER_BEZEICHNUNG    CHAR(50 CHAR)         not null,
   REGAL_NR             NUMBER(15)            not null,
   POSITION_NR          NUMBER(15)            not null,
   POSTEN_ID            NUMBER(38)
)
   partition by list (LAGER_BEZEICHNUNG) automatic
/
```

Partitionen für die einzelnen Lager (*partition by list*) automatisch erzeugt (*automatic*) - da meist mit einem konkreten Lager auf LAGERPOSITION zugreifen wird, kann von Partition-Pruning profitiert werden, nur ein Bruchteil der Tabelle muss durchsucht werden
außerdem könnten in einer Realanwendung die Datenbankzugriffe der einzelnen Lager besser parallelisiert werden

```
/*==============================================================*/
/* Index: IDX_LAGERPOSITION_POSTEN_GLO                          */
/*==============================================================*/
create unique index IDX_LAGERPOSITION_POSTEN_GLO on LAGERPOSITION (
   POSTEN_ID ASC
)
global
/

/*==============================================================*/
/* Index: IDX_LAGERPOSITION_ID_LOC                              */
/*==============================================================*/
create unique index IDX_LAGERPOSITION_ID_LOC on LAGERPOSITION (
   ID ASC
)
local
/
```

Globaler Sekundärindex auf die Belegung der Position *POSTEN_id*, somit kann die Position eines *POSTEN* effizienter ermittelt werden, da statt eines Table-Scans nun gezielt indiziert werden kann

Globaler Index auf den Primärschlüssel *ID*, neben dem automatisch vom System erzeugten globalen Primärindex, somit kann eine ID in einem konkreten LAGER gesucht werden (sofern gegeben), nur ein Bruchteil des Umfangs des globalen Index wird durchsucht

SQL – Abfragen (I)

alle persönlichen Jobs einer Person

- Subselects als Common-Table-Expressions
- (Outer) Joins
- group by
- Gruppenfunktionen

```
with raum_subslct as (
    select
        raum.id as RAUM_id,
        rtrim(replace(gebaeude.name, '-', ' ')) || ' - ' || rtrim(replace(raum.bezeichnung, '-', ' ')) as voller_name
    from GEBAEUDE gebaeude
    join RAUM raum
        on raum.gebaeude_standort_x = gebaeude.standort_x
        and raum.gebaeude_standort_y = gebaeude.standort_y
),
chef_subslct as (
    select
        chef.id as JOB_id,
        rtrim(name) || ' (ID: ' || pers.id || ')' as name
    from JOB chef
    join JOBVERGABE verg
        on verg.id = chef.VERGABE_id
    join PERSON pers
        on pers.id = verg.PERSON_id
),
arbeitszeit_subslct as (
    select
        zeit.JOB_id as JOB_id,
        sum((zeit.schicht_ende - zeit.schicht_beginn) * 24) as stunden_gesamt,
        min(zeit.datum) as erster_tag,
        max(zeit.datum) as letzter_tag
    from ARBEITSZEIT zeit
    group by JOB_id
),
missionsgruppe_subslct as (
    select
        gruppe.id as GRUPPE_id,
        rtrim(mission.titel) || '(' || rtrim(gruppe.titel) || ')' as voller_titel
        from MISSION mission
        join MISSIONSGRUPPE gruppe
            on gruppe.MISSION_id = mission.id
)
select
    rtrim(job.titel) as "Job",
    treffpunkt.voller_name as "Treffpunkt",
    chef.name as "Chef",
    case
        when job.befristet = 0
            then job.gehalt
        else
            (job.gehalt) / months_between(zeiten.letzter_tag, zeiten.erster_tag)
    end as "monatliches Gehalt",
    case
        when abs(nvl(zeiten.erster_tag, vergabe.datum) - SYSDATE) < 365
            then to_char(nvl(zeiten.erster_tag, vergabe.datum), 'DD.MM.')
        else
            to_char(nvl(zeiten.erster_tag, vergabe.datum), 'DD.MM.YYYY')
    end as "erster Tag",
    case
        when job.befristet = 0
            then null
        when abs(zeiten.letzter_tag - SYSDATE) < 365
            then to_char(zeiten.letzter_tag, 'DD.MM.')
        else
            to_char(zeiten.letzter_tag, 'DD.MM.YYYY')
    end as "letzter Tag",
    round(
        zeiten.stunden_gesamt / ((zeiten.letzter_tag - zeiten.erster_tag + 1)  / 7)
    ) as "Wochenstunden",
    missionsgruppe.voller_titel as "Mission",
    to_char(rtrim(job.auftrag)) as "Aufgabe"
from JOB job
join JOBVERGABE vergabe
    on vergabe.id = job.VERGABE_id
join raum_subslct treffpunkt
    on treffpunkt.RAUM_id = job.RAUM_id
left outer join chef_subslct chef
    on chef.JOB_id = job.CHEF_id
left outer join arbeitszeit_subslct zeiten
    on zeiten.JOB_id = job.id
left outer join missionsgruppe_subslct missionsgruppe
    on missionsgruppe.GRUPPE_id = job.GRUPPE_id
where
    (
        (befristet = 1 and zeiten.letzter_tag >= SYSDATE)
        or (befristet = 0 and erledigt = 0)
    )
    and vergabe.PERSON_id = 761
/
```

SQL – Abfragen (2)

alle Inhalte einer Bestellung (in Form einer Rechnung)

- Subselects als Common-Table-Expressions
- (Outer) Joins
- Union All
- Window-Funktionen (sum(preis) over …)

```
with produkt_subslct as (
    select
        posten.id as id,
        posten.kaufpreis as preis,
        nvl(posten.anzahl, 1) as anzahl,
        posten.gewicht as gewicht,
        rtrim(coalesce(geraet.bezeichnung, ressource.bezeichnung)) as bezeichnung
    from POSTEN posten
    left outer join GERAET geraet
        on geraet.id = posten.GERAET_id
    left outer join RESSOURCE ressource
        on ressource.id = posten.RESSOURCE_id
)
select
    typ as "Typ",
    produkt_id as "Angebot ID",
    produkt_name as "Produkt",
    anzahl as "Anzahl",
    gewicht as "Gewicht",
    preis as "Preis",
    sum(preis) over (order by produkt_id) as "Summe"
from (
    select
        'Produkte' as typ,
        produkt.id as produkt_id,
        produkt.bezeichnung as produkt_name,
        bestellposten.anzahl * produkt.anzahl as anzahl,
        produkt.gewicht as gewicht,
        produkt.preis as preis
    from BESTELLUNG bestellposten
    join LIEFERBUCHUNG buchung
        on buchung.id = bestellposten.BUCHUNG_id
    join produkt_subslct produkt
        on produkt.id = bestellposten.POSTEN_id
    where buchung.id = 21
union all
    select
        'Transporte' as typ,
        persbef.id as produkt_id,
        case
            when persbef.richtung = 0 then
                'Hinflug - ' || rtrim(person.name)
            else
                'Rückflug - ' || rtrim(person.name)
        end as produkt_name,
        null as anzahl,
        (persbef.gepack_gewicht + person.gewicht) as gewicht,
        null as preis
    from PERSONENBEFOERDERUNG persbef
    join LIEFERBUCHUNG buchung
        on buchung.id = persbef.BUCHUNG_id
    join PERSON person
        on person.id = persbef.PERSON_id
    where buchung.id = 21
union all
    select
        'Übersicht' as typ,
        null as produkt_id,
        'Lieferkosten' as produkt_name,
        null as anzahl,
        null as gewicht,
        lieferung.lieferkosten as preis
    from LIEFERBUCHUNG buchung
    join LIEFERANGEBOT lieferung
        on lieferung.id = buchung.ANGEBOT_id
    where buchung.id = 21
)
order by "Angebot ID"
/
```

SQL – Abfragen (3)

Statistik zum Ressourcenverbrauch der letzten 7 Wochen

- Joins
- group by rollup
- Gruppenfunktionen
- grouping

```
clear columns
clear breaks
column "Produkt" format a20
column "Kategorie" format a20
break on "Zeitraum" skip 1 on "Kategorie"

select
    'KW ' || to_char(ausgabe.datum, 'YY') || '/' || ((trunc(ausgabe.datum, 'iw') - trunc(trunc(ausgabe.datum, 'YYYY'), 'iw')) / 7) as "Zeitraum",
    coalesce(ressource.typ, 'gesamt') as "Kategorie",
    rtrim(ressource.bezeichnung) as "Produkt",
    sum(ausgabe.anzahl) as "Vebrauch gesamt"
from AUSGABE ausgabe
join RESSOURCE ressource
    on ressource.id = ausgabe.RESSOURCE_id
where (ausgabe.datum >= (SYSDATE - 48))
group by rollup (
    'KW ' || to_char(ausgabe.datum, 'YY') || '/' || ((trunc(ausgabe.datum, 'iw') - trunc(trunc(ausgabe.datum, 'YYYY'), 'iw')) / 7),
    ressource.typ,
    ressource.bezeichnung
)
order by
    "Zeitraum" nulls last,
    grouping(ressource.typ),
    "Kategorie" nulls last,
    "Produkt" asc nulls first
/
```

SQL – Abfragen (4)

Freie Joboptionen für eine Person (entsprechend der Qualifkationen)

- Subselects als Common-Table-Expressions
- (Outer) Joins
- group by
- Gruppenfunktionen
- Subselect mit Anti-Join (where not exists (select …))

```
with raum_subslct as (
    select
        raum.id as RAUM_id,
        rtrim(replace(gebaeude.name, '-', ' ')) || ' - ' || rtrim(replace(raum.bezeichnung, '-', ' ')) as voller_name
    from GEBAEUDE gebaeude
    join RAUM raum
        on raum.gebaeude_standort_x = gebaeude.standort_x
        and raum.gebaeude_standort_y = gebaeude.standort_y
),
termine_subsclt as (
    select
        zeit.JOB_id as JOB_id,
        sum((zeit.schicht_ende - zeit.schicht_beginn) * 24) as stunden_gesamt,
        min(zeit.datum) as erster_tag,
        max(zeit.datum) as letzter_tag
    from ARBEITSZEIT zeit
    group by JOB_id
),
schichtzeiten_subslct as (
    select distinct
        JOB_id,
        to_char(schicht_beginn, 'HH24:MI') as beginn,
        to_char(schicht_ende, 'HH24:MI') as ende
    from ARBEITSZEIT
),
chef_subslct as (
    select
        chef.id as JOB_id,
        rtrim(name) || ' (ID: ' || pers.id || ')' as name
    from JOB chef
    join JOBVERGABE verg on verg.id = chef.VERGABE_id
    join PERSON pers on pers.id = verg.PERSON_id
)
select distinct
    id as "ID",
    rtrim(job.titel) as "Job",
    treffpunkt.voller_name as "Treffpunkt",
    nvl(chef.name, 'offene Stelle') as "Chef",
    case
        when job.befristet = 0
            then job.gehalt
        else
            (job.gehalt) / months_between(termine.letzter_tag, termine.erster_tag)
    end as "monatliches Gehalt",
    case
        when abs(termine.erster_tag - SYSDATE) < 365
            then to_char(termine.erster_tag, 'DD.MM.')
        else
            to_char(termine.erster_tag, 'DD.MM.YYYY')
    end as "erster Tag",
    case
        when job.befristet = 0
            then null
        when abs(termine.letzter_tag - SYSDATE) < 365
            then to_char(termine.letzter_tag, 'DD.MM.')
        else
            to_char(termine.letzter_tag, 'DD.MM.YYYY')
    end as "letzter Tag",
    round(
        termine.stunden_gesamt / ((termine.letzter_tag - termine.erster_tag + 1)  / 7)
    ) as "Wochenstunden",
    zeiten.beginn as "Schichtbeginn",
    zeiten.ende as "Schichtende",
    to_char(rtrim(job.auftrag)) as "Aufgabe"
from JOB job
join raum_subslct treffpunkt
    on treffpunkt.RAUM_id = job.RAUM_id
left outer join termine_subsclt termine
    on termine.JOB_id = job.id
left outer join chef_subslct chef
    on chef.JOB_id = job.CHEF_id
left outer join schichtzeiten_subslct zeiten
    on zeiten.JOB_id = job.id
where
    not exists (
        select 1
        from JOBQUALIFIKATION jobquali
        where jobquali.JOB_id = job.id
        and jobquali.QUALI_bezeichnung not in (
            select quali.QUALI_bezeichnung
            from QUALIFIKATIONSNACHWEIS quali
            join NACHWEIS nach on quali.NACHWEIS_id = nach.id
            where nach.PERSON_id = 500
        )
    )
    and job.GRUPPE_id is null
    and job.VERGABE_id is null
order by "Job", zeiten.beginn
/
```