Oracle-SQL Dokumentation

Quelle: ORACLE für Profis
Von Wolfgang D. Misgeld
Datum: 23.08.2003

Inhaltsverzeichnis:

1. Umsetzen des Datenbankentwurfs

1.1. Erstellen einer Tabelle „CREATE TABLE“

Erstellt die Struktur einer DB-Tabelle.

```
CREATE TABLE [schema.]<tablellenname>
(<spaltenname> <Datentype>, …) [NULL | NOT NULL];
```

Datentypen:

Datentyp	Beschreibung	Genauere Beschreibung
CHAR(n)	Zeichenfolge	Speicherung von Zeichenketten, Maximallänge 240 Zeichen.
DATE	Datumswerte	Speichert nicht nur das Datum, sondern auch einen Zeitwert. Das gesamte Format hat das Aussehen: jjjjmmtthhmmss
LONG	Zeichenfolge	Für lange Zeichenketten bis maximal 65536 Zeichen. Es ist nur eine Spalte dieses Typs erlaubt. Sie kann nicht indiziert oder als Such- bzw. Ordnungsbegriff in Abfragen benutzt werden. Auch dürfen sie nicht in Tabellen vorkommen, die in geschachtelten Abfragen angesprochen werden.
NUMBER[(g[,d])]	Dezimalzahl	Dient zur Aufnahme numerischer Werte. Die Gesamtstellenanzahl wird mit g (Genauigkeit) angegeben, während d (Dezimalstellen) die Anzahl der Nachkommastellen spezifiziert. Die Definition NUMBER (5,5) legt den Wertebereich von: -0,99999 bis + 0,99999 fest.
RAW(n)	Binärdaten	RAW kann bis 240 Byte Binärdaten, wie sie z.B. von Grafikprogrammen erzeugt werden, speichern.
LONG RAW	Binärdaten	Erlaub die Aufnahme von Binärdaten, bis 65535 Byte.

- Option „NULL“: In der entsprechenden Spalte sind NULL-Werte zulässig.
- Option „NOT NULL“: In der entsprechenden Spalte müssen immer Werte stehen.

1.2. Löschen einer Tabelle „DROP TABLE“

Dieser Befehl dient dazu eine bereits erstellen Table wieder zu löschen.

```
DROP TABLE [Urheber.]<tabelenname>;
```

Normalerweise kann nur der Urheber die Tabelle löschen. Lediglich Benutzer mit DBA-Privilegien können Tabellen anderer Benutzer löschen. Tabellendaten und Indizes werden gelöscht. Views bleiben erhalten, sind jedoch nicht mehr funktionfähig.

1.3. Definition von Synonymen „CREATE SYNOYM“

Synonyme sind Alternativnamen für bestehende Tabellen oder Views.

```
CREATE [PUBLIC] SYNONYM <Synonymname> ON [Urheber] <Tabellenname>;
```

- Option: PUBLIC: Wird PUBLIC definiert kann jeder DB-Benutzer sich in anderen SQL-Befehlen darauf beziehen.

```
DROP [PUBLIC] SYNONYM <Synonymname>;
```

```
RENAME <alter Tabellenname> TO <neuer Tabellenname>
```

1.4. Anlegen von Primär- und Sekundärkeys „CREATE INDEX“

Indizes dienen zur Beschleunigung von Suchoperationen in Tabellen bei der Abfrage mittels SELECT oder beim Ändern mittels UPDATE, aber auch beim LÖSCHEN mit DELETE und zwar immer dort, wo ein Vergleichskriterium in der WHERE-Klausel dieser Befehle angegeben wird.
Indizes können sehr viel Speicher verbrauchen. Daher ist es ratsam, sich genau zu überlegen, für welches Attribut bzw. für welche Tabelle ein Index benötigt wird. Ein Index stellt die Eindeutigkeit eines Tupels her.
Jede Tabelle hat naturgemäß nur einen Primärindex, auch wenn es mehrere Attribute von eindeutigen Charakter geben sollte. Prinzipiell können jedoch beliebig viele Indizes (Sekundärindizes) vereinbart werden.

```
CREATE [UNIQUE] INDEX <Indexname> ON [Urheber] <Tabellenname>
(<Spaltenname> [ASC | DESC]) [COMPRESS | NOCOMPRESS] [SYSSORT |
NOSYSSORT] [ROWS = n] [PCTFREE = 20 | n]
```

```
DROP INDEX [Urheber] <Indexname> [ON <Tabellenname>]
```

- UNIQUE: Erlaubt die Feststellung, dass der Index eindeutig sein muss und ist für den Primärschlüssel immer anzuwenden. ORACLE überwacht diese Option.
- <Indexname>: Der Indexname wird nach den gleichen Regeln wie der Tabellenname gebildet.
- <Urheber>: Ist die UserID des Benutzers, der die Tabelle angelegt hat.
- <Tabellenname>: Die Basistabelle, für die der Index angelegt werden soll.
- <Spaltenname>: Die Spalten der Tabelle, die den Index bilden, wird mehr als eine Spalte angegeben, so handelt es sich um einen zusammengesetzten Schlüssel

- ASC | DESC: Mit dieser Option wird die Sortierreihenfolge der Indexwerte festgelegt. ASC (ascending) bedeutet aufsteigend „z.B. A nach Z“.
- COMPRESS | NOCOMPRESS: Mit der Option COMPRESS werden die Indexwerte verdichtet in der Tabelle gespeichert. Durch NOCOMPRESS kann es daher zu schnellern Zugriffszeiten kommen.
- SYSSORT | NOSYSSORT: SYSSORT bestimmt, dass zum Erstellen des Indexes die Standardsortierfunktion benutzt wird.
- ROWS: Gibt die ungefähre Anzahl der Zeilen an, die indiziert werden müssen, wodurch SYSSORT optimiert wird.
- PCTFREE: Erlaub die Reservierung von Speicher für Indexzugänge. ORACLE reserviert standardmäßig für n 20 % eines Speicherblocks (1 Block, auch page genannt, = 1024 bis 4096 Byte, je nach Betriebssystem), die zunächst frei bleiben. Ein größerer Wert für zu weniger Aufsplittungen des Indexes, was sich vorteilhaft für die Verarbeitungsgeschwindigkeit auswirkt.

1.5. Modifizieren von Datenbanken „ALTER TABLE“

Datenbanken die bereits erstellt worden sind, können im Nachhinein auch wieder verändert werden. Es kann zum Beispiel vorkommen, dass neue Spalten eingefügt werden müssen oder bestehende Attribute hinsichtlich ihres Datentyps oder Spaltengröße verändert werden müssen.

```
ALTER TABLE [<Urheber>] <Tabellenname> ADD | MODIFY (<Spaltenname>
<Datentyp> [NULL | NOT NULL], …)
```

- ADD: Fügt an die schon bestehenden Spalten eine oder mehrere neu an. Der Datentyp wird genauso wie bei der CREATE TABLE definiert. NOT NULL kann nur angegeben werden, wenn die Tabelle noch leer ist.
- MODIFY: Ändert den Datentyp der angegebenen Spalte, allerdings nur, wenn sie keine Daten enthält, sondern in allen Zeilen nur NULL-Werte hat.

1.6. Tabellenstruktur dokumentieren „COMMENT“

Um sich jederzeit über das Datenbankschema, also über sie definierten Tabellen und ihre Struktur, informieren zu können, können Informationen aus den Systemtabellen bzw. –views, dem sogenannten Katalog abgerufen werden. Dort besteht die Möglichkeit, Kommentare zu den definierten Tabellen und ihren Spalten abzulegen.

```
COMMENT ON TABLE | COLUMN <Tabellenname>.<Spaltenname> |
<Viewname>.<Spaltenname> IS ‘<Kommentar>
```

- <Kommentar>: Er besteht aus einem beliebigen Text. Seine Maximallänge beträgt 240 Zeichen innerhalb der Apostrophe. Er wird in die Spalte REMARKS der Systemviews CATALOG, SYSCATALOG, COL, COLUMNS oder SYSCOLUMNS eingespeichert. Der Kommentar kann wieder gelöscht werden, indem ein Nullstring eingetragen wird.

2. Views

Views sind ausgewählte Sichten auf Tabellen. Bei der Definition einer View wird keine neue Tabelle mit Daten angelegt, sondern nur eine Beschreibung gespeichert. Folgende Vorteile haben Views:

- Die Informationen von Tabellen können den Benutzern individuell an seine Tätigkeit zur Verfügung gestellt werden.
- Komplizierte Abfragen können mittels Views bereits vordefiniert werden.
- Aus Sicherheitsgründen heraus können den Benutzern nur bestimmte Daten angezeigt werden.
- Views gewährleisten besondere Datenunabhängigkeit im Hinblick aus Datenbankanwendungen. Selbst wenn das Schema einer realen Tabelle völlig verändert werden muss, ist es nicht notwendig, die Programme, die solche Anwendungen betreffen, neu zu erstellen.

2.1. Views erstellen „CREATE VIEW“

Views können erst erstellt werden, wenn die dazu gehörige Tabelle existiert. Daten müssen jedoch nicht gespeichert sein.

```
CREATE VIEW <Viewname> [<Aliasname>,…] AS SELECT-Anweisung [WITH CHECK OPTION];
```

```
DROP VIEW [<Urheber>] <Viewname>
```

- SELECT-Anweisung: Hierbei handelt es sich um einen SQL-Select-Befehl, mit dem Daten aus einer Tabelle abgefragt werden. Entsprechend der Formulierung wird der Blickwinkel auf die Daten festgelegt.
- WITH CHECK OPTION: Dies Option bewirkt, dass ein View, der auch für INSERT- oder UPDATE-Funktionen vorgesehen ist, dies nur zulässt, wenn sich die neuen Daten innerhalb des Ausschnitts befinden, der durch eine Abfrage des Views mittels eines SELECT-Befehls in die Ergebnistabelle übernommen wird.

3. Datenschutz

Das Schutzsystem, das mit SQL-Befehlen aufgebaut werden kann, basiert aus drei Ebenen, die den Zugang zu den Informationen unterschiedlich absichert:

- Systemberechtigungen: Der Zugang zum Datenbanksystem selbst ist nur autorisierten Personen gestattet, die sich durch ihre Benutzerkennung, der sog. User-ID, und einem Passwort beim System anmelden müssen. Besondere Berechtigungen sind die Ressourcen- und DBA-Autorität. Letztere erlaubt, alle Operationen und Funktionen auszuführen, ohne dass SQL-Beschränkung auferlegt.
- Berechtigungen für Tabellen und Views: Der Zugriff auf Tabellen und Views eines anderen Benutzers ist solange nicht möglich, solange jener ihn nicht gestattet. Der Zugriff auf eigene Objekte ist natürlich uneingeschränkt möglich.
- Weitergabe von Rechte an Dritte: Erhält ein Benutzer von einem anderen oder DB-Admin ein gewisses Privileg, also ein Zugriffsrecht für eine oder mehrere

Datenbanken, kann er dieses seinerseits zunächst nicht anderen erteilen, es sein denn, dass ihm das Recht auf Weitergabe ausdrücklich eingeräumt wird.

3.1. Einrichten von ORACLE-Usern „GRANT“

Die Bekanntgabe eines neuen DB-Benutzers erfolgt mit dem GRANT-Befehl:

```
GRANT CONNECT | RESOURCE | DBA, … TO <UserID,…> [IDENTIFIED BY
<Pasßwort>, … ]
```

- CONNECT: Definiert einen Benutzer der keine besonderen Privilegien hat, außer der Berechtigung, sich beim RDBMS anzumelden. Diese Benutzer dürfen zwar ihr Passwort ändern, allerdings keine Tabellen, Views etc. anlegen. Sie können lediglich auf die Daten anderer Benutzer zugreifen, sofern diese freigegeben sind.
- RESOURCE: Kennzeichnet einen DB-Anwender, der die Berechtigung hat, Ressourcen zu verwalten. Beinhaltet automatisch auch CONNECT. Die Benutzer mit diesem Recht dürfen eigene Tabellen, Views etc. anlegen.
- DBA: Ist die höchste Stufe der Berechtigung. ORACLE verlangt zusätzlich CONNECT in Kombination mit DBA. Vollzugriff auf alle Anwenderdatenbanken und mit gewissen Einschränkungen auf die Systemtabellen.
- UserID: Benutzerbezeichnung, unter dem sich die Anwender beim RDBMS anmelden müssen. Maximal 30 Zeichen lang.
- IDENTIFY BY: Diese Klausel muss nur angegeben werden, wenn ein neuer Anwender eingerichtet wird oder wenn ein Benutzer sein Passwort ändern möchte. Für die Änderung der Zugangsrechte eines existierenden Benutzers, z.B. für die Erweiterung auf die DBA-Rechte, kann diese Option entfallen, wenn nicht auch das Passwort gleichzeitig geändert werden soll.
- Passwort: Zusammen mit der Klausel „IDENTIFY BY“ wird für jeden Benutzer ein Schutzwort vergeben, das nur er kennt. Ohne ein korrekte Passworteingabe beim Anmelden gewährt ORACLE keinen Zugang zum System.

Die Berechtigungen CONNECT, RESOURCE, DBA haben Vorrang vor den Privilegien, die auf Tabellenebene vergeben werden können.

3.2. Zugriffsrechte definieren „GRANT“

Privilegien der Benutzer müssen erst erteilt werden, nachdem sie mit GRANT angelegt wurden.

```
GRANT [ALL | ALTER | DELETE | INDEX | INSERT | SELECT | UPDATE
(<Spaltenname>, …) ON <Tabellenname> | <Viewname> TO <UserID>,… | PUBLIC
[WITH GRANT OPTION
```

- All: Der User hat einen uneingeschränkten Zugriff zum angegeben Objekt. ALL schließ alle nachfolgenden Privilegien ein.
- ALTER: Erlaubt Änderungen des Schemas der angegebenen Datenbank in dem Maße, wie es durch den Befehl ALTER TABLE möglich ist. Es können auch neue Spalten hinzugefügt werden.
- DELETE. Gestattet, Zeilen aus dem Objekt zu löschen.
- INDEX: Macht es dem Nehmer möglich, eigene Indizes für die angegebene Tabelle zu definieren und wieder zu löschen.
- INSERT: Erlaubt das Einspeichern von neuen Zeilen in das Objekt.

- SELECT: Lässt Abfragen auf das Objekt zu.
- UPDATE: Ohne Spaltenangabe ermöglicht dieses Privileg die Änderungen der Daten in alle Spalten und Zeilen des Objekts. Die Angabe einer Spaltenliste schränkt die Änderungsmöglichkeiten auf die genannten Attribute in allen Zeilen des Objekts ein.
- PUBLIC: Gibt die aufgeführten Autorisierungen an alle dem System bekannten Benutzer weiter, so dass diese zu den entsprechenden Objekten Zugang haben.
- WITH GRANT OPTION: Der Geber gestattet es von Autorisierungen es dem Nehmer, seinerseits die gleichen Rechte die er erhält, an Dritte weiterzugeben.

3.3. Entziehung von Zugriffsrechten „REVOKE“

Sowohl die Systemrechte als auch die Autorisierungen auf Tabellen können mit dem Befehl REVOKE wieder entzogen werden. Die Syntax entspricht weitgehend dem GRANT-Befehl. Es genügt einem Benutzer das CONNECT-Recht zu entziehen, um ihn mit DBA- oder RESOURCE-Autorität den Zugang zum System zu sperren. Hat der Benutzer Tabellen, Indizies und Views angelegt, bleiben sie weiter bestehen. Sie können allerdings nur noch vom DBA verwaltet werden.

REVOKE Format 1: Nur von Usern mit DBA-Recht zu verwenden:

```
REVOKE CONNECT | RESOURCE | DBA FROM <UserID>,…
```

REVOKE Format 2:

```
REVOKE ALTER | DELETE | INDEX | INSERT | SELECT | UPDATE ALL
(<Spaltenname>,…) ON <Tabellenname> | <Viewname> FROM <UserID> | PUBLIC
```

4. Datenverwaltung

Die Datenverwaltung wird mit den Befehlen der Datenmanipulationssprache (DML) durchgeführt.

- INSERT
- UPDATE
- DELETE
- SELECT

4.1. Laden von Daten „INSERT“

Zur besseren Eingabe von Daten sollten Masken mit ORACLE-Forms entwickelt werden. Sollen nur relativ wenige Zeilen erstmalig erfasst werden, ohne dass spezielle Erfassungsprogramme mit entsprechenden Eingabemasken zur Verfügung stehen, geschieht dies mit dem SQL-Befehl INSERT, der die Daten zeilenweise in die vorgesehene Tabelle einträgt.

```
INSERT INTO [<Urheber>] <Tabellenname> | <Viewname> (<Spaltenname>,…) [SELECT-
Anweisung] VALUES (Wert,…);
```

Beispiel:

Eingabe von zwei Werten in zwei unterschiedliche Spalten.
INSERT INTO tabellenname (spaltenname1,spaltenname2) VALUES (‚Erika’,’31.10.1999);

Eingabe einer kompletten Zeile.
INSERT INTO tabellenname VALUES (10, 1, ‘Erika’, ‘Ehefrau’, NULL);

Kopie einer Tabelle erstellen:
INSERT INTO Neu_Tabelle SELECT * FROM Alt_Tabelle;

Eine physische Sortierung einer Tabelle ist direkt nach der Eingabe nicht mehr möglich.

*4.2. Massendaten importieren mit SQL*Loader*

Das Laden erfolgt mittels zweier Dateien:

- Steuerdatei: in der der Ladeaufruf und die Satzbeschreibung der Datendatei formuliert werden.
- Datendatei: die die zu speichernden Daten enthält.

Aufrufsyntax

```
SQLLOAD [USERID=] <UserID>/<Passwort> [CONTROL=] <Steuerdatei> [LOG=]
<Logdatei> [BAD=] <Fehldatei> [SKIP=n] [LOAD=m]
```

- CONTROL=: Name der Steuerdatei
- LOG=: Name der Logdatei
- BAD=: Name der Logdatei für Fehlermeldungen
- SKIP=: Erlaubt das Überspringen der ersten n Datensätze in der Datendatei, so dass der Ladevorgang mit Satz n + 1 beginnt.
- LOAD=: Angabe wie viel Datensätze maximal (= m) geladen werden sollen und zwar beginnend mit dem ersten Satz nach SKIP.

Beispiel einer Steuerdatei: „mitarbeiter.ctl“

```
LOAD DATA
INFILE mitarbeiter.dat
RECLEN 10
INTO TABLE mitarbeiter REPLACE
(       MA_NR       POSITION    (05:07)       INTEGER EXTERNAL,
        VNAME       POSITION    (09:23)       CHAR,
        NNAME       POSITION    (25:34)       CHAR          )
```

Standardmäßig nimmt Oracle eine Satzlänge von 80 für die Datendatei an. Bei längeren Sätzen ist in der Steuerungsdatei die Option RECLEN anzugeben, wobei zu beachten ist, dass Dateien, welche mit einem Editor erstellt wurden, am Ende der Zeilen noch 2 Zeichen besitzen können (CR LF). Daher mit RECLEN immer 2 Zeichen mehr angeben.

4.3. Verändern von einzelnen Attributen „UPDATE“

Mit einer UPDATE-Anweisung kann ein Attributwert einer Zeile fortgeschrieben werden, es können jedoch auch mehrere Attribute betroffen sein.

```
UPDATE [<Urheber>] <Tabellenname | Viewname> [<Bezugsname>] SET
<Spaltename>=<Wertausdruck> | NULL,…|
(<Spaltenname,…>) = (<Unterabfrage>),…
WHERE <Suchbedingung>;
```

Beispiel: Die Prämie aller Mitarbeiter wird auf 250 gesetzt.
UPDATE mitarbeiter SET praemie = 250;

4.4. Löschen von Zeilen „DELETE“

Das Löschen von Datensätzen wird durch ORACLE immer mengenorientiert ausgeführt.

```
DELETE FROM [<Urheber>] <Tabellename | Viewname> [<Bezugsname>] [WEHRE –
Suchbedingung]
```

Beispiel: Alle Daten der Tabelle „kostenstelle“ sollen gelöscht werden.

DELETE FROM kostenstelle;

Beispiel: Einen bestimmten Datensatz löschen.
DELETE FROM mitarbeiter WHERE ma_nr = 150;

4.5. SQL-Abfragen mit „SELECT“

Der SELECT-Befehl dient zum Abfragen von Informationen aus der Datenbank.

```
SELECT [ALL | DISTINCT] * | <Spaltenname> | <Ausdruck> [<Aliasname>],… FROM
[<Urheber>] <Tabellenname> | <Viewname> [WHERE Suchbedingung]
```

- Ausdruck: In der Select-Liste können Ausdrücke folgender Kategorien auftreten:
  → Konstanten (numerisch oder String)
  → arithmetische Ausdrücke
  → Funktionsausdrücke
  → Datumsausdrücke
  → Systemvariablen
- WHERE: Definition einer Suchbedingung, um nur Zeilen zu selektieren, die ein bestimmtes Suchkriterium als Datum enthalten.

Beispiele:

Liste aller Stellenbezeichnungen und die zugehörige Tätigkeitsnummer. Entfernung aller Wiederholzeilen:
SELECT DISTINCT bezeichnung, t_nr FROM stelle;

Die Spalten der Ergebnistabelle sollen für einen Bericht durch senkrechte Linien deutlicher voneinander getrennt werden. Außerdem sind durch Aliasnamen neue Spaltenbenennungen vorzunehmen:
SELECT ‚|' „|", kst_nr kostenstelle, ‚|' „|", bezeichnung, ‚|' „|", plan_stellen, ‚|' „|", ma_nr mitarbeiter, ‚|' „|" FROM kostenstelle;

4.5.1. Suchbedingung „WHERE"

Ausgabe einer Untermenge durch einen SELECT-Befehl:

WHERE <Operand 1> Operator <Operand 2>

Operanden können sein:

- Spaltennamen von Tabellen oder Views, die in der Form-Klausel genannt werden.
- Numerische oder aphanumerische Konstanten (Literale). Konstanten müssen dem Datentyp der Spalte entsprechen, mit der sie verglichen werden.
- Arithmetische Ausdrücke: Spalten vom Typ NUMBER, DECIMAL, FLOAT, INTEGER oder SMALLINT. Als arithmetische Operatoren können verwendet werden: +, -, *, /, (),
- Systemvariablen: z.B.: LEVEL, NULL, SYSDATE, UID, USER etc.
- Unterabfragen: =, >, <, >=, >=, <>, !=, ^=, Achtung, es wird auch zwischen Groß- und Kleinschreibung unterschieden.

4.5.2. Prüfung, ob der Spaltenwert in einer Werteliste vorkommt „IN":

WHERE <Spaltenname> [NOT] IN (<Ausdruck>,…)

Beispiel: Gesucht werden die Daten von Kostenstellen, die nicht die Kostenstellennummern 1000, 1200 oder 1300 aufweisen.
SELECT st_nr, bezeichnung, kst_nr FROM stelle WHERE kst_nr NOT IN (1000, 1010, 1200, 1300);

4.5.3. Vergleich mit einem Wertebereich „BETWENN" und „AND"

WHERE <Spaltename 1> [NOT] BETWEEN <Spaltenname 2> | <Literal 1> | <arithmetischer Ausdruck> AND <Spaltenname 3> | <Literal 2> | <arithmetischer Ausdruck>

4.5.4. Suchen nach einer Übereinstimmung „LIKE“:

Die Werte einer CHAR-Spalte können mittels des Operanten LIKE nach ähnlichen Mustern durchsucht werden. Dazu ist das Muster als Maske mit Globalsymbolen oder Platzhalter vorzugeben. Die Globalsymbole sind das Prozentzeichen „%“ (Platz für alle vorkommenden Zeichenkombinationen) und der Unterstrich „_“ (Platz an einer bestimmten Zeichenposition für jedes auftretende Zeichen).

WHERE <Spaltenname> [NOT] LIKE <Muster>

4.5.5. Funktionen in SELECT-Anweisungen:

Die eingebauten Funktionen, die in ORACLE verfügbar sind, lassen sich grob in folgende Kategorien einteilen:

- Spalten- oder Gruppenfunktionen:
  → AVG: Mittelwert
  → COUNT: Anzahl
  → MAX: Maximum
  → MIN: Minimum
  → STDDEV: Standardabweichung
  → SUM: Summe
  → VARIANCE: Varianz
- Skalarfunktionen: Skalarfunktionen sind solche Funktionen, die jeweils eine oder mehrere Zeilen bearbeiten im Gegensatz zu den Spalten- oder Gruppenfunktionen, die zur Bearbeitung von Spalten bzw. Gruppen herangezogen werden.
- Datums-Funktionen
- Arithmetische Funktionen:
  → ABS (Wert): Absolutwert
  → CEIL (Wert): Kleinste Ganzzahl, die größer oder gleich <Wert> ist
  → FLOOR (Wert): Größte Ganzzahl, die kleiner oder gleich <Wert> ist
  → MOD (Wert-1, Wert-2): Divisionsrest der Berechnung von <Wert1> / <Wert2>
  → POWER (Wert,n): <Wert> potenziert mit <n>. <n> muss ganzzahlig sein.
  → ROUND(Wert[,{0|n}]): Wert kaufmännisch gerundet auf <n> Dezimalstellen
  → SIGN(Wert>: Liefert einen Wahrheitswert
- Stringfunktionen:
  → ASCII (text): Liefert den ASCII-Code des 1. Zeichens von „text“.
  → CHR(n): Liefert das Zeichen, das dem ASCII-Dezimalwert „n“ entspricht.
  → INITCAP (text): Der 1. Buchstabe eines jeden Wortes in „text“ wird zu einem Großbuchstaben umgewandelt.
  → INSTR (text-1, text-2[,n[,m]]): Liefert die Stelle in „text-1“ relativ zum Anfang, an der „text-2“ beim m-ten Auftreten gefunden wird. Gesucht wird ab der n-ten Stelle.
  → LENGTH(text): Ermittelt die Anzahl Zeichen in „text“.
  → LOWER(text): Der gesamte Text wird in Kleinbuchstaben umgewandelt.
  → LPAD(text-1,n[,text-2]): Fügt links von „text-1 n-mal als Füllzeichen „text-2“ ein. Fehlt „text-2, werden Blanks eingefügt.
  → LTRIM(text-1,[_|text-2]): Eliminiert die 1. Zeichen links in „text-1“, die mit „text-2“ übereinstimmen. Fehlt „text-2“ wird 1 Blank unterstellt.
  → RPAD(text-1,n[,text-2]): Fügt rechts von text-1 n-mal als Füllzeichen text-2 ein. Fehlt „text-2, werden Blanks eingefügt.

→ RTRIM(text-1,[_|text-2]): Eliminiert die letzten Zeichen rechts in text-1, die mit text-2 übereinstimmen. Fehlt text-2, wird 1 Blank untergestellt.
SOUNDEX(text): Erlaubt phonischen Vergleich von Texten.
→ SUBSTR(text,m[,n]): Löst aus text ab der Position m eine Zeichenkette heraus, deren Zeichenzahl mit n bestimmt werden kann.
→ TRANSLATE(Ausdruck,von,nach): Ermöglicht den Ersatz von Zeichen in Ausdruck durch Zeichen in „nach“, die in „von“ auf gleicher Stelle wie in „nach“ vorkommen.
→ UPPER(text): Wandelt alle Zeichen in „text“ in Großbuchstaben um.
→ USERENV(text): Liefert Informationen über den Benutzer, die gewöhnlich für das Audit-Kontrollsystem benötigt werden. Für text kann folgendes eingesetzt werden:
ENTRYID: Nummer der Systemanmeldung
SESSIONID: Nummer der Sitzung
TERMINAL: symbolischer Name des Terminals

- Konvertier-Funktionen: Mit Hilfe von Konvertierfunktionen können Ausdrücke in andere Formate umgewandelt werden. Dabei kann es sich um numerische oder String-Attribute handeln oder arithmetische Ausdrücke.
  → NVL(Ausdruck,Wert): Wandelt einen NULL-Wert, der im „Ausdruck“ auftritt, in „Wert“ um, wobei Letzterer numerisch, alphanumerisch oder vom Typ DATE sein kann.
  → TO_CHAR (Ausdruck[,’Format’]): Führt für einen numerischen Ausdruck eine Druckaufbereitung durch
  → TO_NUMBER(Ausdruck): Erzeugt aus einem Stringausdruck, der eine Zahl enthält, einen numerischen Wert, der für eine artithmetische Operation dienen kann.

```
AVG | MAX | MIN | STDDEV | SUM | VARIANCE ([ALL | DISTINCT] <Spaltenname | Ausdruck>)
```

Oder

```
COUNT (* | <Spaltenname> | ALL | DISTINCT)
```

4.5.6. Ordnen von Daten durch Gruppenbildung „GROUP BY“

Oftmals werden Gruppenergebnisse gewollt, weil Einzelposten nicht interessieren. Mit der GROUP BY-Klausel lässt sich auf einfache Weise eine Gruppierung von Daten erreichen, die gleiche Werte in einer oder mehr Spalten aufweisen, wobei auch virtuelle Spalten mit einbezogen werden können:

```
SELECT [ALL | DISTINCT] * FROM <Tabellenname | Viewname> WHERE <Suchbedingung> GROUP BY <Spaltenname,… | Ausdruck,…> [HAVING <Gruppensuchbedingung>
```

- HAVING: Wird verwendet, wenn über Gruppen selektiert werden soll.
- WHERE: wird verwendet wenn nicht über Gruppen selektiert werden soll.

Beispiel:
Zeige Beruf, Durchschnittsgehalt und die Anzahl der berücksichtigten Einzelposten für die Berufsgruppe „Kaufmann“, deren Durchschnittsgehalt kleiner als 3900 DM ist.

```
SELECT ausb_beruf, AVG(gehalt), COUNT(*)
        FROM mitarbeiter
        WHERE ausb_beruf LIKE ‚Kaufm%’
        GROUP BY ausb_beruf
        HAVING AVG(gehalt) < 3900;
```

Ausgabe:

Ausb_beruf	AVG(gehalt)	COUNT(*)
Bank-Kaufmann	3100	2
Dipl.-Kaufmann	3100	1
Industrie-Kaufmann	3711	3
Kaufmann	3205	4
Technischer Kaufmann	3800	1

4.5.7. Ordnen von Daten durch Sortieren „ORDER BY“

Die Reihenfolge der Zeilen einer Tabelle sind per Definition bzgl. ihren Attributen nicht sortiert. Gewöhnlich werden die Daten in Folge, wie sie eingegeben wurden sortiert. Die Sortierung der Ergebnisstabelle einer Abfrage nach einer oder mehreren Spalten wird durch die „ORDER BY“-Klausel sortiert.

```
SELECT [ALL | DISTINCT] <Spaltenname1> | <Ausdruck1>,… [Aliasname]
FROM [<Urheber>] <Tabellenname> | <Viewname>
[WHERE-Suchbedingung]
[GROUP BY <Spaltenname2> | <Ausdruck2] [HAVING Gruppensuchbedingung]
[ORDER BY <Spaltenname3> | <Ausdruck3> | <Referenznummer> [ASC | DESC],…];
```

- Spaltenname3: Die Attribute, die sortiert werden sollen, müssen nach dem ANSI/ISO-Standard durch die Spaltenliste der SELECT-Anweisung in die Ergebnisliste projiziert werden. Die Sortierhierachie nimmt nach rechts ab.
- Ausdruck3: Diese Alternative ist vom ANSI/ISO-Standard zwar nicht vorgesehen, wird von ORACLE jedoch unterstützt. Als Ausdrücke dürfen hier arithmetische oder Gruppenfunktionen auftreten, die in der Select-Liste vorkommen.
- Referenznummer: Wenn in der SELECT-Anweisung keine Spaltenliste aufgeführt ist, weil das Globalzeichen * für alle Spalten verwendet wird oder wenn ein Ausdruck als Sortierkriterium dienen soll, ist eine Benennung in der ORDER BY Klausel nicht möglich. Statt dessen muss die Referenznummer der Spalte entsprechend ihrer Anordnung in die Ergebnistabelle herangezogen werden.
- ASC | DESC: Mit dieser Option wird die Sortierreihenfolge der Attributwerte festgelegt. ASC ➔ aufsteigend; DESC ➔ absteigend

4.5.8. Verknüpfung mehrerer Tabellen „JOIN“

Das Verfahren, das die Zusammenführung von Spalten verschiedener Tabellen in einer Ergebnistabelle ermöglicht, wird als JOIN bezeichnet. Dabei handelt es sich um eine Erweiterung der FROM-Klausel.

SELECT [ALL | DISTINCT] [<Urheber>.<Tabellenname>.<Spaltenname> [<Aliasname>] FROM [<Urheber>.]<Tabellenname> <Bezugsname>,… [WHERE Join-Bedingung]

Beispiel:

SELECT a_1, b_1, a_2, b_2 FROM test_1, test_2 WHERE a_1 = a_2;

Ohne WHERE-Klausel warden die Spalten als kartesisches Produkt ausgegeben. Daher benutzt man immer eine WHERE-Klausel.

Eine andere Möglichkeit, um die Ausgabe von gleichen Zeilen zu unterdrücken, ist die Angabe von BREAK.

```
BREAK ON nummer ON name ON einkommen
SELECT b.ma_nr nummer, b.nname name, b.gehalt+NVL(b.praemie,0) einkommen,
        a.ma_nr, a.nname, a.gehalt+NVL(praemie,0) endgehalt
        FROM mitarbeiter a, mitarbeiter b
        WHERE a.gehalt+NVL(a.praemie,0) > b.gehalt+NVL(b.pramie,0)
        AND b.nname = ‘Hagedorn’
        ORDER BY 6;
```

- Equi-Join: Die Join-Bedingung wird in diesem Fall durch einen Gleichheitsoperator ausgedrückt. Zur Verbesserung der Ausführungszeit sollten die Spalten mit einem Index belegt werden.

  Beispiel:
  SELECT nname, bezeichnung FROM mitarbeiter a, stelle b WHERE a.st_nr = b.st_nr;

- Outer-Equi-Join: Es warden alle Werte angezeigt, bei denen die Bedingung nicht zutrifft. Das wird erreicht durch den ORACLE-Operator (+).

  Beispiel:
  SELECT b.ma_nr, b.vname, vname, nname, a.vname
  FROM angehoergige a, mitarbeiter b
  WHERE a.ma_nr(+) = b.ma_nr
  AND lf_nr(+) = 1
  ORDER BY 4;

- Non-Equi-Joins: Statt dem "=" Operator warden die anderen Operatoren herangezogen: <, >, <=, >=, <>

  Beispiel:
  SELECT a.ma_nr, a.nname, a.gehalt, SUM(b.gehalt) „KUM.GEHALT"
  FROM mitarbeiter a, mitarbeiter b
  WHERE a.ma_nr >= b.ma_nr
  GROUP BY a.ma_nr, a.nname, a.gehalt;

4.5.9. Vereinigung von Tabellen "UNION"

Hiermit lassen sich horizontale Untermengen (Zeilen) aus mehreren Tabellen zusammenführen. Im Gegensatz zu JOIN (Verknüpfung von Tabellen, wird dieser Vorgang als (Vereinigung oder Zusammenführen von Tabellen) bezeichnet.

```
SELECT-Anweisung UNION | INTERSECT | MINUS SELECT-ANWEISUNG…
ORDER BY <Referenznummer> [ASC | DESC],…
```

- UNION: Jede SELECT-Anweisung für sich ergibt ein Teilergebnis, das der zu erwartenden Einzelergebnistabelle entspricht. Der UNION-Operator führt diese Teilergebnisse zusammen, so dass eine einzige Ergebnistabelle entsteht.
- INTERSECT: Führt aus den beiden durch den Operator vereinigten Datenbanken nur die Zeilen zusammen, die beiden Tabellen vorkommen (= Durchschnittsmenge)
- MINUS: Liefert nur die Zeilen aus der ersten der beiden durch diesen Operator verbundenen Abfrage, die nicht aus durch die zweite selektiert werden (=Diferenzmenge)
- ORDER BY: Darf sich nur auf die letzte SELECT-Anweisung anschließen, da sie sich auf die endgültige Ergebnistabelle bezieht, die aus den zusammengeführten Tabellen der Einzelabfragen gebildet wird.

4.5.10. Unterabfragen „Subqueries"

Unterabfragen sind solche Abfragen, bei denen in der Suchbedingung einer SELECT-Anweisung als rechtsseitiger Vergleichsoperator nicht ein Attribut oder Literal auftritt, sondern eine weitere SELECT-Anweisung. Diese liefert ein Ergebnis, das der übergeordneten Abfrage als Vergleichswert dient.

Regeln zur Bildung von Subqueries:

- Eine Subquery wird immer als rechtsseitiger Term einer Vergleichs- oder EXITS-Bedingung ausgedrückt, wobei sie in Klammern zu setzen ist.
- In der Select-Liste der Unterabfrage müssen die Ausdrücke – Spaltenname oder Spaltenfunktionen- bzw. mathematische Ausdrücke – hinsichtlich Anzahl, Reihenfolge und Datentypen denen der Vergleichsattribute entsprechen. Eine Ausnahme bildet der Operator EXISTS
- Wenn ein Vergleich nur mit den Operatoren =, <, >, >=, <=, <> gebildet wird, muss die Unterabfrage so formuliert werden, dass sie nur einen Wert (eine einzige Zeile) als Ergebnis liefert.
- Bei der Verwendung der Operatoren IN, ANY, ALL und EXISTS kann das Ergebnis der Subquery mehrzeilig sein.

- Die Klausel ODER BY ist innerhalb der Subquery nicht zulässig, sondern erst im Anschluß daran als Fortsetzung der Hauptabfrage.
- Innerhalb des Abfrageblocks einer Unterabfrage ist die UNION-, INTERSECT- bzw. MINUS-Operation nicht erlaubt.

Beispiel: Wie lauten die Mitarbeiternummern und Namen der Personen, die auf der gleichen Kostenstelle arbeiten wie Mitarbeiter Nr. 460

```
SELECT ma_nr, nname
        FROM mitarbeiter
        WHERE st_kst =
                (SELECT st_kst FROM ma_nr = 460);
```

Subquery mit IN: Formulierung von mehrzeiligen Ergebnissen
Beispiel: Welche Mitarbeiter sind verheiratet?

```
SELECT ma_nr, vname, nname
        FROM mitarbeiter
        WHERE ma_nr IN
                (SELECT ma_nr
                FROM angehoerige
                WHERE vw_grd LIKE ‚Ehe%’);
```

Welche Mitarbeiter haben Kinder?

```
SELECT ma_nr, vname, nname
        FROM mitarbeiter
        WHERE ma_nr NOT IN
                (SELECT ma_nr FROM angehoerige
                        WHERE vw_grd = ‚Sohn’
                        OR vw = ‚Tochter’)
                ORDER BY ma_nr;
```

Subquery mit ALL und ANY: Formulierung von Vergleichen außer „=“
Diese Abfrage liefert mehrzeilige Ergebnisse.

Beispiel: Welche Mitarbeiter haben keine Angehörige?
Die Prüfung auf „<> ALL“ ist identisch mit „NOT IN“

```
SELECT ma_nr, vname, nname
        FROM mitarbeiter WHERE ma_nr <> ALL
        (SELECT ma_nr FROM angehoerige)
        ORDER BY 1;
```

oder

```
SELECT ma_nr, vname, nname
        FROM mitarbeiter
        WHERE NOT ma_nr = ANY
                (SELECT ma_nr FROM angehoerige)
        ORDER BY 1;
```

Subquery mit EXISTS: Formulierung ob Werte überhaupt existieren.

Es wird eine Ergebnistabelle produziert, unabhängig davon, welche Werte in ihr vorkommen.

Beispiel: Zeige alle Mitarbeiter, für die Angehörige gespeichert sind.

```
SELECT ma_nr, vname, nname
    FROM mitarbeiter ma
        WHERE EXISTS
        (SELECT * FROM angehoerige an
                WHERE ma.ma_nr = an.ma_nr)
        ORDER BY 1;
```

Beispiel: Welche Arbeitsstellen im Unternehmen sind nicht besetzt?

```
SELECT * FROM stelle a
    WHERE NOT EXISTS
    (SELECT * FROM mitarbeiter b
            WHERE a.st_nr = b.st_nr);
```

In der Unterabfrage muss immer ein Vergleichskriterium zwischen der Subquery und der Hauptabfrage angegeben warden.

5. Regeln mit CONSTRAINTS erstellen

Oracle verwendet CONSTRAINTS, um ungültige Dateneinträge in Tabellen zu verhindern. Folgende CONSTRAINT-Typen sind zulässig:

- NOT NULL (Gibt an, dass diese Spalte keine Nullwerte enthalten darf)
- UNIQUE (Eindeutig)
- PRIMARY KEY (Identifiziert eindeutig jede Zeile der Tabelle)
- FOREIGN KEY (Verweis auf eine externe Tabelle)
- CHECK (Gibt eine Bedingung an, die wahr sein muss)

ORACLE generiert CONSTRAINTS mit dem Namen „SYS_Cn-Formats). CONSTRAINTS können bei dem Erstellen oder nach dem Erstellen von Tabellen aufgebaut werden. Man kann sie auf Spalten und Tabellenebene definieren.

5.1. CONSTRAINTS definieren

```
CREATE TABLE [schema.]table (column datatype [DEFAULT expr] …
[column_constraint])
```

- schema: entspricht dem Eigentümernamen
- table: Ist der Name der Tabelle
- DEFAULT expr.: Gibt einen Standardwert an, falls in der INSERT-Anweisung ein Wert weggelassen wurde.
- Column: Ist der Name der Spalte
- Datatype: Datentyp und die Länge der Spalte
- Column_constraint: Ist ein Inegritäts-Constraint als Teil der Spaltendefinition
- Table_constraint: Ist ein Integritäts-Constraint als Teil der Tabellendefinition

CONSTRAINTS auf Spaltenebene:

```
Column [CONSTRAINT constraint_name] constraint_type,
```

Constraint auf Tabellenebene

```
Column,…
        [CONSTRAINT constraint_name] constraint_type
        (column,…),
```

Beispiel:

```
CREATE TABLE emp(
                Empno NUMBER(4),
                Ename VARCHAR2(10),
                …
                deptno NUMBER(2) NOT NULL,
                CONSTRAINT emp_empno_pk
                        PRIMARY KEY (empno));
```

5.2. Constraints hinzufügen

```
ALTER TABLE table ADD [CONSTRAINT constraint] type (column)
```

- table: Ist der Name der Tabelle
- constraint: Ist der Name des Constraint
- type: Ist der Constraint-Type
- column: Ist der Name der vom Constraint betroffenen Spalte

Beispiel:

```
ALTER TABLE emp ADD CONSTRAINT emp_mgr_fk FOREIGN KEY(mgr)
REFERENCES emp (empno);
```

5.3. Constraints löschen

```
ALTER TABLE table DROP [CONSTRAINT constraint_name]
```

- table: Ist der Name der Tabelle
- constraint: Ist der Name des Constraint

5.4. Constraints deaktivieren

```
ALTER TABLE table DISABLE CONSTRAINT constraint [CASCADE];
```

Die CASCADE-Klausel schaltet abhängige Integritäts-Constraints aus.

5.5. Constraints aktivieren

```
ALTER TABLE table ENABLE CONSTRAINT constraint
```

- Wenn ein Constraint eingeschaltet wird, gilt dieses für alle Daten in der Tabelle. Alle Daten in der Tabelle müssen auf das Constraint passen.
- Wenn ein UNIQUE- oder PRIMARY-KEY eingeschaltet wird, wird der Index hierzu automatisch erstellt.
- Die ENABLE-Klausel kann sowohl in CREATE und ALTER-Klauseln verwendet werden.

5.6. Constraints anzeigen

```
SELECT constraint_name, constraint_type, search_condition FROM user_constraints
WHERE table_name = ‘<Tabellename>’
```

5.7. Spaltennamen und ihre Constraints anzeigen

```
SELECT constraint_name, column_name FROM user_cons_columns WHERE table_name =
‚EMP’
```