ANHANG A

ISO 9075 SQL Syntax und ORACLE Erweiterungen

Nachfolgend werden nur die wichtigsten Befehle mit den am häufigsten verwendeten Optionen aufgeführt. Zur vollständigen Definition wird auf den Standard **SQL:1999 (ISO/IEC 9075)** und die SQL-Bücher unserer Bibliothek verwiesen.

SQL ist eine nichtprozedurale, abbildungsorientierte Sprache zur Manipulation und Abfrage von relationalen (SQL-) Datenbanken.
Sie enthält alle Sprachelemente, die für Datenbankdefinition, Abfragen und Änderungsoperationen einer relationalen Datenbank benötigt werden. Die Sprache basiert auf dem Relationenkalkül, wobei Prädikate und Quantoren durch eine leicht erlernbare, verbale Syntax ersetzt werden. Sie eignet sich auch deshalb für 'ad hoc' Abfragen, weil in einer SQL-Phrase nur das 'Was', nicht jedoch das 'Wie' beschrieben wird, d. h. Operationen, wie man die Daten bereitstellt (Prozeduren), werden nicht angegeben. Man nennt SQL aus diesem Grund auch nichtprozedural oder deskriptiv. SQL wirkt auf Tabellen bzw. Relationen. Es können Manipulationen auf Spalten und Zeilen oder auf Verknüpfungen (Abbildungen) von Spalten und Zeilen auf einer Tabelle (bzw. dem kartesischen Produkt von Tabellen) durchgeführt werden.

Beispiel : Tabelle emp (liegt allen Beispielen des Anhangs A zugrunde)

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7329	SMITH	CLERK	7902	12-DEC-90	800	300	20
7499	ALLEN	SALESMAN	7698	20-FEB-91	1'600	300	30
7521	WARD	SALESMAN	7698	22-FEB-91	1'250	500	30
7566	JONES	MANAGER	7839	02-APR-91	2'975		20
7654	MARTIN	SALESMAN	7698	28-SEP-91	1'250	1'40	30
7698	BLAKE	MANAGER	7839	01-MAY-91	2'850		30
7782	CLARK	MANAGER	7839	09-JUN-91	2'450		10
7788	SCOTT	ANALYST	7566	09-DEC-82	3'000		20
7839	KING	PRESIDENT		17-NOV-81	5'000		10
7844	TURNER	SALESMAN	7698	08-SEP-91	1'500	0	30
7876	ADAMS	CLERK	7788	12-JAN-83	1'100		20
7900	JAMES	CLERK	7698	03-DEC-91	950		30
7902	FORD	ANALYST	7566	03-DEC-91	3'000		20
7934	WILLER	CLERK	7782	23-JAN-92	1'300		10

SQL-Abfrage :

```
select ename, job, sal          ← Auswahl Spalten
from emp                        ← Tabellenname
where empno > 7500 and empno < 7600 ;   ← Auswahl Zeilen
```

(Die Eingabe ist formatfrei und wird mit Semikolon abgeschlossen)

Ergebnis :

ENAME	JOB	SAL
WARD	SALESMAN	1250
JONES	MANAGER	2975

Notation der Syntax :
Die Darstellung orientiert sich an der Backus-Naur-Form (BNF). Die linke Seite von ::= wird durch ihre rechte Seite definiert. SQL-Sprachelemente sind fett dargestellt, Groß- oder Kleinschreibung ist irrelevant, <name> ist ein Platzhalter für einen Bezeichner oder Ausdruck, [...] bezeichnen optionale Teile, | bezeichnet eine Auswahl, ... bezeichnen optionale Repetitionen. ORACLE-spezifische Erweiterungen sind durch *Kursivschrift* gekennzeichnet.

A1) Anlegen einer Tabelle :

create table <table> (<col1> <typ1> [<kb1>],<col2> <typ2> [<kb2>],...,[<tkb>])
[***tablespace <tspace> [pctfree <n>]***] | [***cluster <cluster> (<colx>, <coly>,...)***];

wobei <colx> ::= Spaltenname x, (x = 1,2,3,...)

<table> ::= Tabellenanme

<typx> ::= **char[(n)] | char varying(n) | varchar2(n) | date | decimal[(n[,m])] | float[(n)] | integer | smallint |** ***number(n[,m])*** **| numeric[(n[,m])] | real | double precision**
mit x, n, m = 1,2,3,..., n = max. Gesamtstellenzahl, m = Nachkommastellen

<kbx> ::= **default** <expr> **| not NULL | unique | primary key |**
references <table>[(<col>)][**on delete cascade**] | **check** (<cond>)
(Konsistenzbedingungen für Spalten)

<cond> ::= logischer Ausdruck , z. B. <colx> rel <valx>
mit <rel> ::= < | > | <= | >= | != | = | <> | **like** und
<valx> ::= Spaltenwert (auch Platzhalter % und _)

<expr> ::= '<text>' | <cfunction>('<text>') | <num-expr>

<cfunction> ::= Funktionen zur Textmanipulation (lower, upper, lpad, rtrim, length, to_date)

<num-expr> ::= numerischer Wert oder Ausdruck mit *, /, + oder - .

<text> ::= Folge alphanumerischer Zeichen

<tbk> ::= **unique** (<col1>[, <col2>, ...]) | **primary key** (<col1>[, <col2>, ...]) |
foreign key (<col1>[, <col2>, ...]) **references** <table>[(<col1>[, <col2>, ...])]
[**on delete cascade**] | **check** (<cond>)
(Konsistenzbedingungen für die Tabelle)

<tspace> ::= Name der Tablespace-Definition

<n> ::= Prozentwert 0 ... 99

<cluster> ::= Name der Cluster-Definition

Der Datentyp **char** ist eine Zeichenkette fester Länge, während **char varying** und **varchar2** variable Länge mit maximal n Zeichen besitzen.
Tablespace und Cluster sind ORACLE-spezifische Erweiterungen. Sie werden durch Befehle, die dem '**create table**' ähnlich sind, erzeugt (**create tablespace** ..., **create cluster** ...).
Die Konsistenzbedingung '**not NULL**' bedeutet, daß für jede Zeile ein Spaltenwert vorhanden sein muß.
'**unique**' verlangt die Eindeutigkeit der Werte, '**primary key**' definiert einen Identifikationschlüssel.

```
Beispiel :      create table emp  (empno number(4) primary key,    ename char(10) not NULL,
                                   job char(8),                    mgr number(4) references emp(empno),
                                   hiredate date,                  sal number(7),
                                   comm number(6),                 deptno number(2) references dept_table,
                                   check (sal + comm <= 5000) );
```

A2) Anlegen eines Index

create index <index> **on** <table> (<col1>, <col2>, ...);

<index> ::= Name des Index
Die sonstige Syntax ist analog zu der von A1).

A3) Ändern einer Tabelle :

alter table <table> **add** (<col1> <typ1> [<kb1>],<col2> <typ2>, ...,<tkb>);
alter table <table> **drop** [<col1> <typ1> <kb1> | <tkb>];
oder
alter table <table> ***modify*** (<col1> <typ1> [<kb1>], <col2> <typ2>...,<tkb>);

Die Syntax ist analog zu der von A1).
Änderungen können nur vorgenommen werden, wenn der Dateninhalt dies gestattet. Z.B. können Felddefinitionen nur dann verkleinert werden, wenn der Inhalt der Spalte nicht größer ist als das gewünschte Format. Felder können nur gelöscht werden, wenn sie leer (NULL) sind.

Beispiele : alter table emp add last_company char(10) ;
alter table emp drop primary key;
alter table emp modify (last_company char(14));

A4) Anlegen einer Benutzersicht (view) :

Um eine logische Datenunabhängigkeit von der Tabellendefinition zu erreichen oder um spezielle Benutzersichten zu erzeugen, können Views angelegt werden.

create [or replace] view <vname> [(<alias1>, <alias2>, ...)] **as** <subquery> [**with check option**];

wobei <aliasx> ::= virtueller Spaltenname x, (x = 1,2,3,...)
<vname> ::= Viewname
<subquery> ::= SQL-Query wie unter A7 beschrieben

Ein View kann wie eine zusätzliche (virtuelle) Tabelle gesehen werden, die aber ihre Daten von der realen Tabelle bezieht. Auf einen View können grundsätzlich die gleichen Datenmanipulationen angewandt werden wie auf eine Tabelle. Die ORACLE-Datenbank erzwingt allerdings einige Einschränkungen. Views, die über einen Verbund (join), '**group by**' oder '**distinct**' definiert wurden, erlauben nur '**select**'- Befehle.
'**With check option**' bedeutet, daß **ein 'insert into, update, delete'** nur Daten erzeugen kann, die auch über den View selektiert werden können.

Beispiel : create view empdep10 (MA-Nr, MA-Name, Beruf) as
select empno, ename, job from emp
where deptno = 10;

Beispiel einer Abfrage auf den View :

select * from empdep10; (vgl. A8)

Ergebnis :

MA-Nr	MA-Name	Beruf
7782	CLARK	MANAGER
7839	KING	PRESIDENT
7934	WILLER	CLERK

A5) Daten/Zeile in Tabelle schreiben :

insert into <table> [(<col1>, <col2>, ...)]
values (<val1> , <val2> , ...);
oder
insert into <table> [(<col1>, <col2>, ...)] <subquery>;

wobei <colx> ::= Spaltenname x, (x = 1,2,3,...)
<table> ::= Tabellen- oder View-Name
<valx> ::= numerischer oder alphanumischer Wert entsprechend der Spaltendefinition (Inhalt des Feldes colx := valx), (x = 1,2,3,..)
<subquery> ::= Query wie unter A7)

Die mit der Subquery gefundenen Datensätze müssen zu der Spaltenliste (<col1>, <col2>, ...) in Anzahl Felder, Feldfolge und Datentyp zusammenpassen.

Beispiele :
```
insert into emp (empno, ename, job, mgr,hiredate, sal, deptno)
    values (7369,'SMITH','CLERK',7902,17-DEC-80, 800,20);

insert into emp (empno, ename, job, mgr,hiredate, sal, deptno)
    select distinct * from ext_emp where deptno = 40;
```

Ext_emp muß 7 Felder in der gleichen Reihenfolge mit verträglichen Datentypen besitzen wie die o.g. Spalten der Tabelle emp.

A6) Daten ändern :

update <table> **set** <col1> = <val1>, <col2> = <val2> , ...
[where <cond> <op> <cond> ....];
oder
update <table> **set** (<col1>, <col2>, ...) = (subquery)
[**where** <cond> <op> <cond> ....];

wobei <colx> ::= Spaltenname, (x = 1,2,3,...)

<table> ::= Tabellename

<valx> ::= Spaltenwert (Inhalt für colx), Subquery oder NULL, wenn der Wert frei bleiben soll, x = 1,2,3,..

<cond> ::= logischer Ausdruck , z. B. <colx> rel <valx>
mit rel = < | > | <= | >= | != | =

<op> ::= **and** | **or**

Die Änderung der Daten darf nicht im Widerspruch zu den Konsistenzbedingungen stehen, da sonst der Update abgewiesen wird.

Beispiel :
```
update emp set   job = 'MANAGER',
                 hiredate = 26-FEB-88,
                 comm = comm * 1.2
                 where sal < 10000;
```

Wenn hierbei sal + comm > 5000 erreicht würde, kann der Update nicht durchgeführt werden.

A7) Daten löschen :

delete [**from**] <table> [**where** <cond> <op> <cond> ....];

wobei <cond> ::= logischer Ausdruck , z. B. <colx> <rel> <valx>

<rel> ::= < | > | <= | >= | != | =

<op> ::= **and** | **or**

<table> ::= Tabellenanme

Beispiel : delete from emp
where hiredate = 26-FEB-88 and comm = 20;

A8) Daten suchen und anzeigen (Query):

select [all | distinct] <col1>, <col2>, ... **from** <table1>, <table2>, ...
[**where** <cond> <op1> <cond> .... | **where** <cola> <op2> <subquery>]
[**group by** <colr>, <cols>, ...] | [***start with*** <start>] ***connect by*** <conn>
[**having** <cond> <op1> <cond> ....]
[**union | intersect | minus**] <subquery>]
[**order by** <coln> [DESC], <colm> [**desc**], ...]
[***for update of*** <colx>, coly>, ... [***nowait***]];

wobei <colx> ::= [Tabelle.]Spaltenname_x [Alias] (x = 1,2,3,...)
<tablex> ::= [<user>.] <table_view_x> [Alias] (x = 1,2,3,...)
<user> ::= User-Name (die Selektionsrechte müssen mit **grant** ... erteilt werden)
<table_view> ::= Tabellen- oder View-Name
<cond> ::= logischer Ausdruck , z. B. <colx> <rel> <valx>
<rel> ::= < | > | <= | >= | != | = | <> **| like** (Wertevergleich)
<valx> ::= Spaltenwert (auch Platzhalter % und _)
<op1> ::= **and | or**
<srel> ::= <rel> **all** | <rel> **any** | **[not] in | [not] exists** (Mengenvergleich)
<op2> ::= <srel> | <rel> (Mengen- und Wertevergleich)
<start> ::= Startbedingung z.B. job = 'PRESIDENT' oder subquery
<conn> ::= ***PRIOR*** <cond> (Beziehungsbedingung z.B. PRIOR empno = mgr and sal > comm)
desc ::= absteigende Sortierfolge (aufsteigend ist Default)
distinct ::= gleiche Datenzeilen werden unterdrückt (Default ist **all**)

Ein Spaltenname <colx> kann temporär einen zweiten Namen (Aliasnamen) erhalten, um z.B. bei mehrfachem Vorkommen der Spalte, diese unterscheiden zu können. Anstelle von <col1>,... kann auch * treten, um alle Felder / Spalten auszuwählen. Anstelle von <col1>,... kann auch ein arithmetischer Ausdruck treten, z. B. <col1>*1.15, <col2>/<col3> (Achtung NULL-Werte!). Anstelle von <col1>,... können auch Funktionen verwendet werden, z. B. sum(<col>*1.15), max(<col>/<col>), lpad('Name = ',1) || <col> = Aneinanderhängen von 2 Zeichenketten. Aggregatfunktionen (z.B. avg(<col>) = Mittelwert, count(<col>) = Anzahl Werte) sind in Verbindung mit '**group by**'-Spalten einzusetzen.
Bei Vergleichsoperationen mit einer Subquery ist darauf zu achten, ob die Query einen einzelnen Wert oder eine Menge von Werten erzeugt. Im ersteren Fall kann für <op2> ein Wertevergleich <rel> eingesetzt werden, während im zweiten Fall ein Mengenvergleichsoperator <srel> verwendet werden muß.
Eine hierarchische Suche kann durch ***'start with ... connect by'*** durchgeführt werden.
Mit der Bedingung ***'for update of'*** kann eine Satzsperre angefordert werden. Sind die Daten bereits durch eine andere Transaktion gesperrt, so wird auf die Freigabe gewartet, wenn nicht **'nowait'** angegeben wurde. Eine Sperre stellt sicher, dass die Daten bis zum nächsten **'commit'** oder '**rollback**' nicht durch andere Transaktionen verändert werden.

Beispiele : select * from emp;

Ergebnis : Ausgabe der Tabelle emp (siehe Seite A-1).

```
select job, count(empno), avg(sal) from emp
where deptno = 20
group by job;
```

Ergebnis :

JOB	COUNT(EMPNO)	AVG(SAL)
CLERK	2	950
MANAGER	1	2850
ANALYST	2	3000

```
select ename, sal from emp
where  ename like 'F%' or sal <= (select avg(sal)/2 from emp);
```

Ergebnis :

ENAME	SAL
FORD	3000
SMITH	800
JAMES	950

ORACLE unterstützt hierarchische Abfragen der Form:

```
select lpad('   ',2*(level-1)) || ename, empno, mgr, job
from emp
start with job = 'PRESIDENT' connect by prior empno = mgr;
```

Ergebnis :

ENAME	EMPNO	MGR	JOB
KING	7839		PRESIDENT
JONES	7566	7839	MANAGER
SCOTT	7788	7566	ANALYST
ADAMS	7876	7788	CLERK
FORD	7902	7566	ANALYST
SMITH	7369	7902	CLERK
BLAKE	7698	7839	MANAGER
ALLEN	7499	7698	SALESMAN
WARD	7521	7698	SALESMAN
MARTIN	7654	7698	SALESMAN
TURNER	7844	7698	SALESMAN
JAMES	7900	7698	CLERK
CLARK	7782	7839	MANAGER
WILLER	7934	7782	CLERK

Die Pseudospalte 'level' (siehe A11 Pseudospalten) gibt die Nummer der Hierarchiestufe an, sie kann wie eine normale Tabellenspalte referiert, aber nicht geändert werden. Die Funktion 'lpad' setzt so viele Leerzeichen links vor den Wert von ename, wie durch den Ausdruck 2*(level-1) errechnet wird. Für die 3 Hierarchiestufe erhält man den Wert 4, also 4 Leerzeichen vor dem Namen SCOTT.

A9) Tabelle/View löschen :

drop table <table>;

drop view <vname>;

wobei <table> ::= Tabellenname
<vname> ::= Name der Benutzersicht (view).

Beispiele : drop table emp;

drop view empdep10;

A10) Tabelle für andere Benutzer freigeben :

Eine Tabelle kann anderen Benutzern (auch teilweise) zur Verfügung (Abfrage, Mutation, ...) gestellt werden :

Objektprivilegien :

grant <obj-priv1>, <obj-priv2>, ... **on** <table>
to <user1>, <user2>, ... | **public**
[**with grant option**] ;

Systemprivilegien :

grant <sys-priv1>, <sys-priv2>, ...
to <user1>, <user2>, ... **| public**
[with admin option] ;

wobei <table> ::= Tabellen-, View oder Sequence-Name
<userx> ::= Name des Benutzers (x = 1,2,3,...)
<obj-privx> ::= **all privileges | select | insert | delete | update** [(<col1>, <col2>, ...)]
<sys-privx> ::= **connect | ressource | dba | exp_full_database** | <privilege>
<privilege> ::= z.B. **alter user, backup any table, create sequence, lock any table**, etc.

Beispiele : grant select on table emp to public;

grant select, update (sal, comm) on emp to wi4;

grant backup any table to wi4;

A11) Passwort ändern :

Ein Benutzer kann sein ORACLE-Passwort durch folgenden Befehl ändern :

grant connect to *<user> [**identified by** <passwd>];*

wobei <user> ::= Name des Benutzers
<passwd> ::= neues Passwort

Beispiele : grant connect to wi4_01 identified by geheim;

A12) Sequenzenummern / Pseudospalten / Systemfunktionen:

ORACLE kann systemweit eindeutige Sequenznummern erzeugen, die der Transaktionskontrolle unterliegen.

create sequence <sequ> [**increment by** <incr>][, **start with** <first>][, **cache** <val>];

wobei <sequ> ::= Name der Zahlenfolge
<incr> ::= Inkrementwert der Folge (Default = 1)
<first> ::= Startwert der Folge

Beispiel : create sequence empno
increment by 1 start with 7000;

ORACLE verfügt über eine Anzahl von Pseudospalten, die in SQL-Abfragen verwendet werden können:

CURRVAL ::= gibt den aktuellen Wert der eigenen Sitzung zurück (z.B. auftr_nr.currval)

NEXTVAL ::= gibt den nächsten Wert für die eigene Sitzung zurück und speichert ihn als aktuellen Wert (z.B. select auftr_nr.nextval from dual; dual ist eine Tabelle des Users SYS, die von jedem Benutzer benutzt werden kann.)

LEVEL ::= zeigt die Hierarchiestufe einer Abfrage an, die mit '**start with ... connect by**' formulierten wurde.

ROWID ::= interne Zeilennummer im hexadezimalen Format <block-nr|.<zeilen-nr>.<file-nr>

ROWNUM ::= laufende Nummer für die Ergebniszeilen einer Abfrage. Damit kann die Anzahl Zeilen einer Abfage begrenzt werden.
(z. Bsp.: **select * from** emp **where** rownum < 10; begrenzt die Ausgabe auf 9 Zeilen)

Zusätzlich stehen folgende Systemfunktionen zur Verfügung:

SYSDATE ::= gibt das aktuelle Systemdatum und die Systemzeit aus

USER ::= gibt den ORACLE-Benutzernamen zurück

TERMINAL ::= gibt die Terminalnr. aus

A13) Transaktionskontrolle :

commit [**work**];

Schließt eine Transaktion ab, schreibt die Werte fest und beginnt eine neue Transaktion.

rollback [work][to savepoint <savept>];

Bricht eine Transaktion ab und setzt die bisherigen Änderungen zurück. Mit '**to savepoint**' kann ein Teil einer Transaktion zurückgesetzt werden.

savepoint <savept>;

Erzeugt eine identifizierbare Transaktionsmarke zu der die Transaktion zurückgesetzt werden kann.