

# DB I Herr Waldhorst – Zusammenfassung

## 1) Grundlegende Datentypen

CHAR(n) / CHARACTER(n)	<ul style="list-style-type: none"><li>- Zeichenkette mit genau n Zeichen</li><li>- Aufgefüllt mit Leerzeichen</li></ul>
VARCHAR(n) / CHARACTER VARYING(n)	<ul style="list-style-type: none"><li>- Zeichenkette min max. n Zeichen</li></ul>
INT / INTEGER	<ul style="list-style-type: none"><li>- Ganze Zahl, maschinenabhängig</li></ul>
SMALLINT	<ul style="list-style-type: none"><li>- Teilmenge der ganzen Zahlen, maschinenabhängig</li></ul>
NUMERIC(d,p)	<ul style="list-style-type: none"><li>- Festkommazahl, d Ziffern, davon p hinter dem Dezimalpunkt</li></ul>
REAL / DOUBLE PRECISION	<ul style="list-style-type: none"><li>- Fließkommazahl, Genauigkeit maschinenabhängig</li></ul>
FLOAT(n)	<ul style="list-style-type: none"><li>- Fließkommazahl, mit Genauigkeit (mind.) n Stellen</li></ul>

## 2) Weitere Datentypen

DATE	<ul style="list-style-type: none"><li>- Datum</li></ul>
TIME	<ul style="list-style-type: none"><li>- Zeit mit und ohne Zeitzone</li></ul>
INTERVALL	<ul style="list-style-type: none"><li>- Zeitdauer</li></ul>
BLOB	<ul style="list-style-type: none"><li>- Beliebige binäre Zeichenketten, Größe meist Gigabyte</li></ul>
BOOLEAN	<ul style="list-style-type: none"><li>- TRUE oder FALSE</li></ul>

## 3) Befehle

1. Tabellen erstellen/modifizieren oder löschen	
Tabelle erstellen	<b>CREATE TABLE</b> name ( Attributname Datentyp, Attributname Datentyp, );
Tabellenstruktur prüfen	<b>DESCRIBE</b> DatenTypenBeispiel;
Komplette Tabelle löschen	<b>DROP TABLE</b> R;
Attribut A mit Datentyp D zu Tabelle R hinzufügen	<b>ALTER TABLE</b> R <b>ADD</b> A D;
Attribut A aus Relationsschema R löschen	<b>ALTER TABLE</b> R <b>DROP COLUMN</b> A;
Spalte für Anzahl Mitarbeiter einfügen	<b>ALTER TABLE</b> Abteilung <b>ADD</b> AnzMitarbeiter <b>INTEGER</b> ;
Spalte löschen	<b>ALTER TABLE</b> Abteilung <b>DROP COLUMN</b> AnzMitarbeiter;
komplette Tabelle löschen	<b>DROP TABLE</b> Abteilung;
2. Einfügen von Datensätzen mit INSERT	
3x einfügen einzelner Datensätze	<ul style="list-style-type: none"><li>- <b>INSERT INTO</b> MyRiver <b>VALUES</b> ('Ruhr', 'D', 219 );</li><li>- <b>INSERT INTO</b> MyRiver(Length, Country, Name) <b>VALUES</b> (93, 'D', 'Kinzig');</li><li>- <b>INSERT INTO</b> MyRiver(Name, Country) <b>VALUES</b> ('Alb', 'D');</li></ul>
Einfügen einer SELECT-Anfrage	<b>INSERT INTO</b> MyRiver <b>SELECT DISTINCT</b> Name, Country, Length <b>FROM</b> River <b>JOIN</b> Geo_River <b>ON</b> River.Name=Geo_River.River <b>WHERE</b> Country = 'D';

3. Aktualisieren von Datensätzen mit UPDATE	
<ul style="list-style-type: none"> <li>- Aktualisiert Tabelle R</li> <li>- Prädikat p in WHERE-Klausel wählt aus, welche Daten</li> <li>- Ausdruck A in SET-Klausel gibt an, was geändert werden soll</li> </ul>	<ul style="list-style-type: none"> <li>- <b>UPDATE R SET A WHERE P</b></li> <li>- <b>UPDATE</b> MyRiver     <b>SET</b> Length=51     <b>WHERE</b> Name='Alb';</li> <li>- <b>UPDATE</b> MyRiver     <b>SET</b> Length=Length/1.61     <b>WHERE</b> Country='GB';</li> </ul>
<b>CASE</b> = Um Probleme mit Reihenfolge von Aktualisierungen zu vermeiden, können mehrere Aktualisierungen mit CASE-Schlüsselwort kombiniert werden	<b>UPDATE</b> MyRiver <b>SET</b> Length= <b>CASE</b> <b>WHEN</b> Length <= 440 <b>THEN</b> Length * 1.1 <b>ELSE</b> Length * 1.3 <b>END;</b>
4. Löschen von Datensätzen mit DELETE	
Datensatz für Alp löschen	<b>DELETE FROM</b> MyRiver <b>WHERE</b> Name='Alb';
5. Beschränkungen / Constraints	
Attribut muss einen Wert zugewiesen bekommen	<b>NOT NULL</b>
Jeder Wert eines Attributs / einer Menge von Attributen darf nur einmal vorkommen	<b>UNIQUE</b>  <b>CREATE TABLE</b> MyRiver ( Name <b>VARCHAR</b> (50), Country <b>VARCHAR</b> (4), Length <b>NUMBER NOT NULL</b> , <b>UNIQUE</b> (Name, Country) );
Primärschlüssel (Nicht NULL ( <b>NOT NULL</b> ) und eindeutig ( <b>UNIQUE</b> ))	<b>PRIMARY KEY</b> (
Fremdschlüssel <ul style="list-style-type: none"> <li>- Die Attribute müssen auf den Primärschlüssel von Tabelle S referenzieren</li> <li>- Fremdschlüssel von R kann in keinem Datensatz werte annehmen, die in dieser Kombination in S nicht existieren</li> </ul>	<b>FOREIGN KEY</b> (Ak1, Ak2, ..., Akn) <b>REFERENCES</b> S  <b>CREATE TABLE</b> Mitarbeiter ( Personalnummer <b>NUMERIC</b> (4), Nachname <b>VARCHAR</b> (40), Vorname <b>VARCHAR</b> (30), Funktion <b>VARCHAR</b> (30), Abteilung <b>NUMERIC</b> (3), <b>PRIMARY KEY</b> (Personalnummer), <b>FOREIGN KEY</b> (Abteilung) <b>REFERENCES</b> Abteilung );
6. Sonstiges	
Operator LIKE ermöglicht String-Vergleiche mit Platzhaltern <ul style="list-style-type: none"> <li>- % steht für Substring beliebiger Länge (auch keinem Zeichen!)</li> <li>- _ steht für Substring der Länge 1 (d.h. genau einem Zeichen)</li> </ul>	<b>SELECT</b> Name <b>FROM</b> Country <b>WHERE</b> Name <b>LIKE</b> 'B%';  <b>SELECT</b> Name <b>FROM</b> Country <b>WHERE</b> Name <b>LIKE</b> '___land';
Aggregationen = nehmen eine Sammlung (Menge oder Multimenge von Werten und liefern einen einzigen Wert zurück <ul style="list-style-type: none"> <li>- Anwendung in SELECT Klausel</li> <li>- Ergebnis ist immer eine Relation</li> </ul>	<ul style="list-style-type: none"> <li>- Durchschnitt: <b>AVG</b></li> <li>- Minimum: <b>MIN</b></li> <li>- Maximum: <b>MAX</b></li> <li>- Summe: <b>SUM</b></li> <li>- Anzahl: <b>COUNT</b></li> </ul> <b>SELECT</b> <b>AVG</b> (Length) <b>FROM</b> RIVER <b>WHERE</b> Sea = 'North Sea';
Aggregation mit GROUP BY = Aggregation auf Gruppen von Tupeln ausführen	<b>SELECT</b> <b>AVG</b> (Population), Country <b>FROM</b> City <b>GROUP BY</b> Country;
Aggregation mit HAVING = Bedingungen können für Gruppen anstatt für einzelne Tupel definiert werden	<b>SELECT</b> Country, <b>AVG</b> (Population) <b>AS</b> Average <b>FROM</b> City <b>GROUP BY</b> Country <b>HAVING</b> <b>MAX</b> (Population) > 1000000;
<b>DISTINCT</b> = Duplikate können vor Anwendung der Aggregatsfunktion eliminiert werden	<b>SELECT</b> <b>AVG</b> ( <b>DISTINCT</b> Population) <b>FROM</b> City;

ORDER BY = ermöglicht Sortierung der ausgegebenen Tupel nach Attributen <ul style="list-style-type: none"> <li>- ASC (aufsteigend)</li> <li>- DESC (absteigend)</li> </ul>	SELECT Name, Country FROM City ORDER BY Country ASC, Province DESC, Name ASC;
UNIQUE =Vereinigung (entspricht oder - gibt keine Duplikate) (Ausnahme UNION ALL - doppelte Tupel bleiben stehen)	SELECT Country FROM Geo_Lake UNION SELECT Country FROM Geo_River;
INTERSECT (und) = Schnittmenge (entspricht und)	SELECT Country FROM Geo_Lake INTERSECT SELECT Country FROM Geo_River;
EXCEPT in Oracle MINUS = Mengendifferenz	SELECT Country FROM Geo_Lake MINUS SELECT Country FROM Geo_River;
Mitgliedschaft in Mengen (IN, NOT IN) = In WHERE-Klausel erlaubt Operator IN Test einer Mengen-Mitgliedschaft	SELECT AVG(Population) FROM City WHERE Name IN (ODER NOT IN) (SELECT City.Name FROM City NATURAL JOIN Encompasses WHERE Continent = 'Europe' OR Continent = 'Africa');
Mengenvergleiche (>ALL) = Wert mit einer Menge von Werten zu vergleichen <ul style="list-style-type: none"> <li>- &lt;ALL &lt;SOME</li> <li>- &lt;=ALL &lt;=SOME</li> <li>- =SOME</li> <li>- &gt;=ALL &gt;=SOME</li> <li>- &gt;ALL &gt;SOME</li> <li>- &lt;&gt;ALL &lt;&gt;SOME</li> </ul>	SELECT Name FROM River WHERE Length >ALL (SELECT Length FROM River WHERE SEA = 'North Sea');
WITH-Klausel = Temporäre Relationen können mit WITH-Klausel definiert werden	WITH AvgTab AS (SELECT Country, AVG(Population) AS Average FROM City GROUP BY Country) SELECT MAX(Average) FROM AvgTab;
EXISTS = prüft, ob Ergebnis einer Unterabfrage leer ist (oder mindestens ein Tupel enthält)	SELECT Island, Country FROM Geo_Island WHERE EXISTS (SELECT * FROM Island WHERE Name = Island AND Type = 'volcanic');
<b>7. Sichten / Views = Virtuelle Relation, Nicht gespeichert, sondern immer bei Bedarf neu berechnet</b>	
Einfaches Erstellen von Views (wie TABLES)	CREATE VIEW Professoren AS SELECT Name, Abteilung FROM Dozent;
<b>8. Erweiterte Tabellenerzeugung</b>	
Transaktionen = sind atomar, es werden entweder alle Kommandos ausgeführt oder keins	COMMIT WORK Festschreiben der Kommando-Ergebnisse ROLLBACK WORK Rückgängigmachen der Kommandos
Integritätsbedingungen <ul style="list-style-type: none"> <li>- PRIMARY KEY</li> <li>- NOT NULL</li> <li>- UNIQUE</li> <li>- CHECK (&lt;Prädikat&gt;)</li> </ul>	CREATE TABLE Dozent ( Name VARCHAR(30), Abteilung VARCHAR(30), Gehalt NUMBER, PRIMARY KEY (Name), FOREIGN KEY (Abteilung) REFERENCES Sekretariat, CHECK(Gehalt >= 65000) );
Referentielle Integrität = Fremdschlüsselbedingungen (Foreign Key Constraints) stellen referentielle Integrität (Gültigkeit von Verweisen zwischen Relationen) sicher	CREATE TABLE Dozent ( Name VARCHAR(30), Abteilung VARCHAR(30), Gehalt NUMBER, PRIMARY KEY (Name), FOREIGN KEY (Abteilung) REFERENCES Sekretariat );

Referentielle Integrität = bei Definition der Fremdschlüsselbeziehung sorgen dafür, dass referenzierende Tabelle aktualisiert wird (eins löschen aus einer Tabelle führt auch zum Löschen in anderer Tabelle)	<b>CREATE TABLE</b> Dozent ( Name <b>VARCHAR(30)</b> , Abteilung <b>VARCHAR(30)</b> , Gehalt <b>NUMBER</b> , <b>PRIMARY KEY</b> (Name), <b>FOREIGN KEY</b> (Abteilung) <b>REFERENCES</b> Sekretariat <b>ON DELETE CASCADE</b> Oder <b>ON DELETE SET NULL</b> Oder <b>ON DELETE SET DEFAULT</b> );
Referentielle Integrität und Transaktionen = Fremdschlüsselbedingung kann während Transaktion verletzt sein, muss aber am Ende der Transaktion eingehalten werden	<b>CREATE TABLE</b> Ehepaare ( Name <b>VARCHAR(30)</b> , Partner <b>VARCHAR(30)</b> , <b>PRIMARY KEY</b> (Name), <b>FOREIGN KEY</b> (Partner) <b>REFERENCES</b> Ehepaare(Name) <b>INITIALLY DEFERRED</b> );  <b>INSERT INTO</b> Ehepaare <b>VALUES</b> ('John', 'Mary'); <b>INSERT INTO</b> Ehepaare <b>VALUES</b> ('Mary', 'John'); <b>COMMIT</b> ;
Default-Werte = Pro Attribut können Default-Werte festgelegt werden	<b>CREATE TABLE</b> Dozent ( Name <b>VARCHAR(30)</b> , Abteilung <b>VARCHAR(30)</b> , Gehalt <b>NUMBER DEFAULT</b> 65000, <b>PRIMARY KEY</b> (Name), <b>FOREIGN KEY</b> (Abteilung) <b>REFERENCES</b> Sekretariat);  <b>INSERT INTO</b> Dozent(Name, Abteilung)
Automatisch generierter Primärschlüssel = Erzeugung Primärschlüssel über Sequenz	<b>CREATE TABLE</b> DozentMitId ( DozentenId <b>NUMBER</b> , Name <b>VARCHAR(30)</b> , Abteilung <b>VARCHAR(30)</b> , Gehalt <b>NUMBER DEFAULT</b> 65000, <b>PRIMARY KEY</b> (DozentenId), <b>FOREIGN KEY</b> (Abteilung) <b>REFERENCES</b> Gebaeude ); <b>CREATE SEQUENCE</b> SeqDozentId;  <b>INSERT INTO</b> DozentMitId <b>VALUES</b> ( SeqDozentId.NEXTVAL, 'Einstein', 'Physik', 60000 ); <b>INSERT INTO</b> DozentMitId <b>VALUES</b> ( SeqDozentId.NEXTVAL, 'Hawking', 'Physik', 75000 );
<b>Privilegien</b>	
Vergabe (GRANT)	<u>Beispiel: Vergabe von Abfrage-Privilegien</u> <b>GRANT SELECT ON</b> Dozent <b>TO</b> hinz;  <u>Beispiel: Vergabe von Einfüge-Privilegien</u> <b>GRANT INSERT ON</b> Dozent <b>TO</b> hinz; <b>GRANT INSERT</b> (Abteilung) <b>ON</b> Sekretariat <b>TO</b> hinz, kunz;  <u>Beispiel: Vergabe von Aktualisierungs-Privilegien</u> <b>GRANT UPDATE ON</b> Dozent <b>TO</b> hinz; <b>GRANT UPDATE</b> (Abteilung) <b>ON</b> Sekretariat <b>TO</b> hinz, kunz;  <u>Beispiel: Vergabe von Lösch-Privilegien</u> <b>GRANT DELETE ON</b> Dozent <b>TO</b> hinz, kunz;
Rücknahme (REVOKE)	<b>REVOKE SELECT ON</b> Dozent <b>FROM</b> hinz; <b>REVOKE INSERT ON</b> Dozent <b>FROM</b> hinz; <b>REVOKE UPDATE</b> (Abteilung) <b>ON</b> Sekretariat <b>FROM</b> hinz, kunz;
Rollen	<b>CREATE ROLE</b> Hauspost; <b>CREATE ROLE</b> HrInfo; <b>GRANT SELECT ON</b> Postadressen <b>TO</b> Hauspost; <b>GRANT SELECT, INSERT, UPDATE ON</b> InformatikDozenten <b>TO</b> HrInfo; <b>GRANT</b> Hauspost <b>TO</b> hinz;

	<b>GRANT</b> Hauspost <b>TO</b> HrInfo; <b>GRANT</b> HrInfo <b>TO</b> kunz;
WITH GRANT OPTION = Benutzer darf Privilegien weiter übertragen	<b>GRANT SELECT ON</b> Dozent <b>TO</b> hinz <b>WITH GRANT OPTION</b> ;

## WICHTIG:

### Auswertungsreihenfolge



1. **FROM**-Klausel bestimmt die Relation (ggf. mit kartesischem Produkt, Join)
2. **WHERE**-Klausel filtert die Tupel der Relation aus 1)
3. **GROUP BY**-Klausel bildet Gruppen
4. **HAVING**-Klausel filtert Gruppen
5. **SELECT**-Klausel gibt Attribute der Gruppen aus bzw. berechnet Aggregatsfunktionen