

Prof. DI Dr. Erich Gams

# Datenmanipulation in SQL

Einführung und Anwendung in



informationssysteme htl-wels

To start....

Please, close your laptops



and just



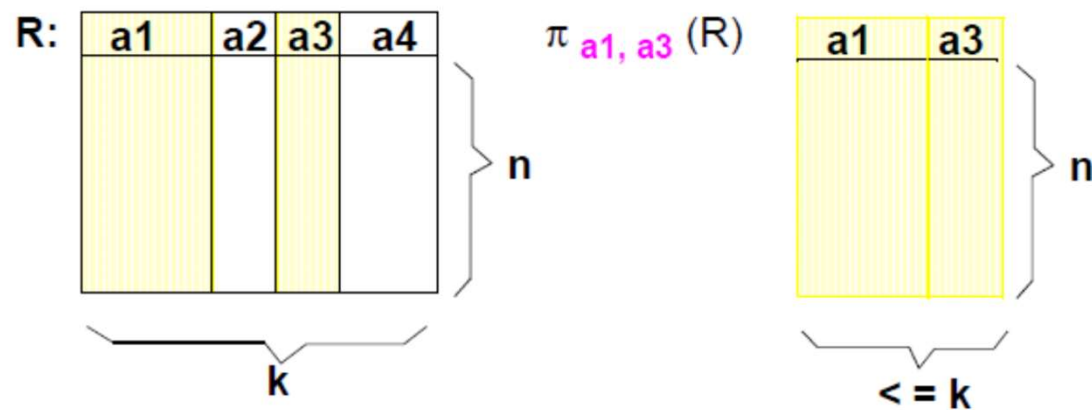
# Übersicht ➡ Was lernen wir?



- › Elementare Operationen auf Relationen
  - Projektion
  - Selektion
  - Patternmatching
  - Verschachtelte Abfragen
- › daraus einfache Anfragen in SQL

# Projektion $\pi$

“vertikale Spalten ausschneiden”:



- aus einer Menge von Attributen nur eine Auswahl anzeigen:  
in SQL:

```
SELECT a1, a3 FROM R
```

- Sonderfall: alle Attribute einer Relation:

```
SELECT * FROM R
```

# Projektion

```
SELECT
    lastname,
    firstname,
    jobtitle
FROM
    employees;
```

Resultatstabelle:

	lastname	firstname	jobtitle
►	Murphy	Diane	President
	Patterson	Mary	VP Sales
	Firelli	Jeff	VP Marketing
	Patterson	William	Sales Manager (APAC)
	Bondur	Gerard	Sale Manager (EMEA)
	Bow	Anthony	Sales Manager (NA)
	Jenninas	Leslie	Sales Rep

	employeeNumb	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
►	1002	Murphy	Diane	x5800	dmurphy@classicmodelcars.com	1	NULL	President
	1056	Patterson	Mary	x4611	mpatterso@classicmodelcars.com	1	1002	VP Sales
	1076	Firelli	Jeff	x9273	jfirelli@classicmodelcars.com	1	1002	VP Marketing

# Sortieren der Ausgabe

## › ORDER BY:

- Das Anfrageergebnis kann mit der Anweisung ORDER BY sortiert dargestellt werden.
- Es kann dabei **nach mehreren Spalten sortiert** werden.

## › ASC und DESC

- Durch den Zusatz **ASC (Aufsteigend)** bzw. **DESC (Absteigend)** wird die Sortierreihenfolge bestimmt.
- Wird keine Sortierreihenfolge angegeben, wird aufsteigend sortiert.

# Sortieren der Ausgabe - Beispiele

```
SELECT
    contactLastname,
    contactFirstname
FROM
    customers
ORDER BY
    contactLastname DESC;
```

```
SELECT
    contactLastname,
    contactFirstname
FROM
    customers
ORDER BY
    contactLastname DESC,
    contactFirstname ASC;
```

- › Anstelle des Spaltennamens, kann auch dessen Positionsnummer (d.h. an welcher Stelle es in der SELECT-Anweisung steht) in der ORDER BY-Klausel angegeben werden.

# Sortieren der Ausgabe nach einer Berechnung

```
SELECT
    orderNumber,
    orderlinenumber,
    quantityOrdered * priceEach
FROM
    orderdetails
ORDER BY
    quantityOrdered * priceEach DESC;
```



# Sortieren nach String-Werten

- › **FIELD** Funktion
- › Sortiert die Spalte *status* nach den nachfolgenden Werten: *In process, On hold* usw.

```
SELECT
    orderNumber,
    status
FROM
    orders
ORDER BY
    FIELD(status,
        'In Process',
        'On Hold',
        'Cancelled',
        'Resolved',
        'Disputed',
        'Shipped');
```

# Distinct

- › Eine Liste aller (**unterschiedlichen=DISTINCT**)  
Nachnamen, absteigend sortiert:

```
SELECT DISTINCT Nachname  
FROM Personal  
ORDER BY 1 DESC;
```

# Umbenennung der Ergebnisspalten AS

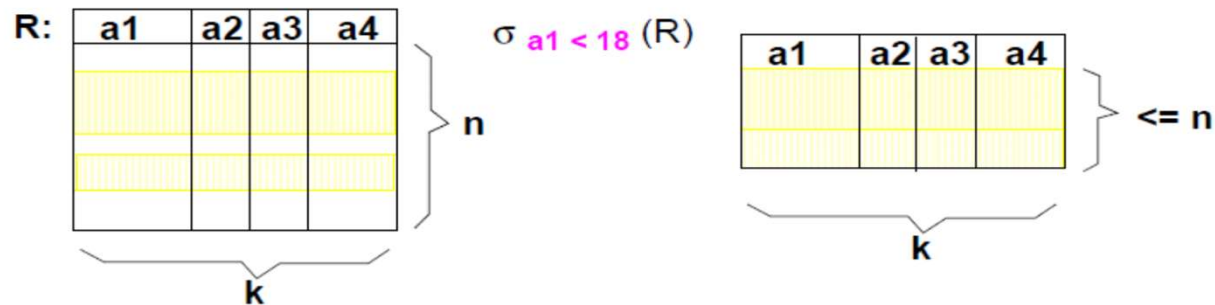
- › Verwendung von berechneten Werten und Spaltenüberschriften:

```
SELECT
    orderNumber,
    orderLineNumber,
    quantityOrdered * priceEach AS subtotal
FROM
    orderdetails
ORDER BY subtotal DESC;
```

	orderNumber	orderLineNumber	subtotal
▶	10403	9	11503.14
	10405	5	11170.52
	10407	2	10723.60

# Selektion $\sigma$

Selektion mit einem Prädikat<sup>1</sup>:



- Das **Selektionsprädikat** formuliert eine Bedingung auf den Werten eines Attributs
- z.B. alle Tupel, deren Wert für a1 kleiner als 18 ist
- in SQL:

SELECT \* FROM R WHERE a1 < 18

<sup>1</sup> Selektionsprädikat heißt auch: Selektionsbedingung

# Selektion

```
SELECT
    lastname,
    firstname,
    jobtitle
FROM
    employees
WHERE
    jobtitle = 'Sales Rep';
```

	lastname	firstname	jobtitle
►	Jennings	Leslie	Sales Rep
	Thompson	Leslie	Sales Rep
	Firrelli	Julie	Sales Rep
	Patterson	Steve	Sales Rep

# Verknüpfung mit AND

```
SELECT
    lastname,
    firstname,
    jobtitle,
    officeCode
FROM
    employees
WHERE
    jobtitle = 'Sales Rep' AND
    officeCode = 1;
```

	lastname	firstname	jobtitle	officeCode
►	Jennings	Leslie	Sales Rep	1
	Thompson	Leslie	Sales Rep	1

# Selektion

- › Um bestimmte Zeilen einer Tabelle auszuwählen, muss angegeben werden, welche Bedingung für diese ausgewählten Zeilen gelten soll.
- › Beispielsweise sollen nur jene Artikel ermittelt werden, von denen mindestens 10 Stück Bestand vorhanden sind.

```
SELECT Artikelnr, Artikelname, Lagerbestand  
FROM Artikel  
WHERE Lagerbestand >= 10  
ORDER BY Lagerbestand DESC;
```

- › Alle Artikel bei denen der Lagerbestand den Mindestbestand unterschritten hat:

```
SELECT Artikelnr, Artikelname, Lagerbestand  
FROM Artikel  
WHERE Lagerbestand < Mindestbestand  
ORDER BY Lagerbestand DESC;
```

# Selektion

- › Eine Liste, die die Bestellmenge für jeden Artikel umfasst. Die Bestellmenge ergibt sich aus der Abweichung von Lagerbestand und Mindestbestand

```
SELECT Artikelnr, Artikelname, Mindestbestand-Lagerbestand AS Bestellmenge  
FROM Artikel  
WHERE Lagerbestand < Mindestbestand  
ORDER BY 3 DESC;
```



# Einfache Abfragen auf eine Relation

- › gib alle Daten(Spalten) über Professoren aus:

```
SELECT * FROM professoren;
```

- › gibt alle Daten(Spalten) von Studenten aus dem 2. Semester aus:

```
SELECT * FROM studenten WHERE semester = 2;
```

- › gib Namen und Matrikelnummer (Spalten) von Studenten des Hauptstudiums aus:

```
SELECT name, matrnr FROM studenten  
WHERE semester <= 4;
```

# Selektionsprädikate

- Vergleichsoperationen <, >, <=, >=, <>

```
SELECT * FROM pruefen WHERE note > 4.0
```

- Mengenvergleiche IN, NOT IN (später mehr ...)

```
SELECT name FROM studenten WHERE semester IN (2,4,6,8);
```

- Wildcards: LIKE

% beliebig viele Zeichen

\_ genau ein Zeichen

```
SELECT * FROM professoren  
WHERE name LIKE 'J%ger'
```

```
SELECT * FROM studenten  
WHERE name LIKE 'A%'
```

```
SELECT * FROM studenten  
WHERE vorname LIKE 'Mari_'
```

# Stringvergleich

## › String-Vergleich

Select \* from "Artikel" where "Artikelname" = 'Scheiben'

Select \* from "Artikel" where "Artikelname" LIKE 'Scheiben'

## › Achtung bei Pattern-Matching immer mit LIKE!

Select \* from "Artikel" where "Artikelname" LIKE '\_cheiben'

Select \* from "Artikel" where "Artikelname" LIKE 'Sch%'

# RegEx Verwendung

```
mysql> SELECT * FROM pet WHERE REGEXP_LIKE(name, '^b');
```

name	owner	species	sex	birth	death
Buffy	Harold	dog	f	1989-05-13	NULL
Bowser	Diane	dog	m	1979-08-31	1995-07-29

# LIMIT

- › Limitieren der Ergebnisse auf 5

```
SELECT DISTINCT  
    state  
FROM  
    customers  
WHERE  
    state IS NOT NULL  
LIMIT 5;
```

# Selektionsprädikate

- › Der logische Ausdruck in der Where-Klausel kann (fast) beliebig komplex gestaltet werden. Für die Abarbeitung des Ausdrucks gelten folgende Rangregeln:

- ›

1)	(	)		
2)	*	/		
3)	+	-		
4)	=	<>	>	<
5)	NOT	AND	OR	

- › Beispiele:

```
.... WHERE Artikelnr >= 25 AND Artikelnr < 50;
.... WHERE Artikelname LIKE 'Schokolade' AND Einzelpreis <= 100;
.... WHERE Einzelpreis BETWEEN 50 AND 240;
      // der kleine Wert vor dem Größeren
.... WHERE Personalnr IN (120, 145, 230, 450);
.... WHERE Artikelname LIKE '%Schokolade%';
.... WHERE Artikelname NOT LIKE '%Schokolade%';
.... WHERE Nachname LIKE '%ma_er%';
```

# Geschachtelte Anfragen



- › **Bedingungen** haben prinzipiell die Struktur
  - **<Attribut> <Operator> <Wert>**
  - z.B. Name=„John“
- › Die Werte können wiederum
  - Attribute,
  - Konstanten oder aber
  - **das Resultat von Unteranfragen sein**
- › d. h. man muss die Werte von Bedingungen **nicht unbedingt fix** definieren, sondern kann sie mittels einer **verschachtelten Anfrage generieren**.
- › **Anfragen können so beliebig tief verschachtelt werden.**

# Geschachtelte Anfragen

- › Es gibt drei Typen von Unteranfragen, welche sich durch ihr Resultat unterscheiden:
- › Unteranfragen die
  - einen Wert zurückgeben (eine Spalte und eine Zeile)
  - eine Zeile zurückgeben
  - mehrere Zeilen zurückgeben



# Geschachtelte Anfragen

- › Ergebnis ist ein Wert oder eine Zeile
  -  Verwende die normalen Vergleichsoperatoren
- › Ergebnis sind mehrere Zeilen
  -  Verwende spezielle Operatoren
  - **IN** sucht ob der Wert in der Unteranfrage vorkommt
  - **<Vergleichsoperator> ALL** die Bedingung muss für alle Zeilen in der Unterabfrage TRUE ergeben
  - **<Vergleichsoperator> ANY (SOME)** die Bedingung muss für mindestens eine Zeile in der Unterabfrage TRUE ergeben

# Beispiele von Mengenoperationen

- › Mengenoperatoren EXISTS, ALL, ANY, SOME, IN, NOT IN
  - IN: Wert muss in der Liste enthalten sein
  - NOT IN: Wert darf nicht in der Liste enthalten sein
  - Kombinationen mit <, >, = möglich.
  - Achtung: Redundanz der Operatoren:
- › < ALL: **kleiner als das Minimum einer Liste, dasselbe ergibt: < ANY**
- › > ANY: **größer als das Maximum einer Liste, dasselbe ergibt: > ALL**
- › = SOME: **gleich mit irgendeinem in der Liste, dasselbe ginge auch mit IN**
- › = ALL: **gleich allen in der Liste**
  - => alle Werte in der Liste müssen gleich sein, sonst wenig Sinn.

# Beispiel geschachtelte Anfragen

```
Select Vorname, Nachname From Kunde  
Where KundenNr IN  
    (Select KundenNr  
     From Abonnement  
     Where Name="Die Presse")
```

Welche Tabellen existieren in der Datenbank?  
Wie schauen die Ergebnistabellen aus?

## ... oder auf der Nordwind DB

Bestellung	( <u>BestellNr</u> , ↑KundenCode, ↑PersonalNr, Bestelldatum, Lieferdatum, Versanddatum, ↑FirmenNr, Frachtkosten, Empfänger, Straße, Ort, Region, PLZ, Bestimmungsland)
Kunde	( <u>KundenCode</u> , Firma, Kontaktperson, Position, Straße, Ort, Region, PLZ, Land, Telefon, Telefax)

```
SELECT "Kunde"."Firma"
FROM nordwind_mdb."Kunde"
WHERE "Kunde"."KundenCode" IN
  (SELECT "Bestellungen"."KundenCode" FROM
    nordwind_mdb."Bestellungen"
   WHERE "Bestellungen"."Bestimmungsland"='Schweiz')
```