Handout
Themenfeld: Datenbanken und SQL
Abschnitt: 08.01.01.DQL_WHERE-Ausdruck

Autor: Thomas Krause
Stand: 14.11.2022 12:03:00

Inhalt

1 Vorbereitung

Für den Vortrag wird verwendet:

- database_hochbau_MUSTER_DB.sql
- database_hochbau_MUSTER_DB_ERM_RM.dia

Installieren Sie bei Bedarf die Datenbank und verschaffen Sie sich nochmals einen Überblick.

2 Verwendung von WHERE: Einführung, Grundlagen und Vertiefung

(1) Die WHERE – Anweisung wird benötigt, um aus allen zur Verfügung stehenden Datensätzen nur die benötigten Datensätze zu selektieren.
(2) WHERE wird immer mit einem logischen Ausdruck verwendet. Dieser logische Ausdruck kann einfach oder komplex sein.

3 Verwendung der Platzhalter % und _ für die Suche in Strings

Beispiele:

Ausgangstabelle:

Tabelle für nachfolgende Abfragen

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M009	0	Örtel	04105	9
M010	1	Stein	04838	12
M021	1	Hahn	04509	10
M024	0	Holzer	04119	9
M113	1	Latte	04381	NULL
M117	1	Fahrland	04512	NULL
M118	0	Putz	04838	NULL

verschiedene Abfragen

Ergebnislisten (verschiedene Beispiele):

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M021	1	Hahn	04509	10
M024	0	Holzer	04119	9

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M021	1	Hahn	04509	10
M113	1	Latte	04381	NULL
M117	1	Fahrland	04512	NULL

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M010	1	Stein	04838	12
M021	1	Hahn	04509	10

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M021	1	Hahn	04509	10
M117	1	Fahrland	04512	NULL

Syntax:

```
SELECT <spaltenliste> FROM <tabelle>
WHERE <spaltenname> LIKE '[<zeichen>]%[<zeichen>]'
```

Erläuterung:

- % steht für **eine beliebige Anzahl** beliebiger Zeichen
- muss immer mit dem Operator LIKE verwendet werden

```
SELECT <spaltenliste> FROM <tabelle>
WHERE <spaltenname> LIKE '[<zeichen>]_[<zeichen>]'
```

Erläuterung:

- _ steht für **ein** beliebiges Zeichen
- muss immer mit dem Operator LIKE verwendet werden

praktische Anwendung:

Ausgangstabelle:

siehe oben

Ergebnislisten:

Beispiel 1:

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M021	1	Hahn	04509	10
M024	0	Holzer	04119	9

```
select * from tbl_mitarbeiter
where mitarbeitername like 'h%';
```

Beschreibung:

- alle Mitarbeiter, die an erster Stelle in mitarbeitername den Buchstaben h bzw. H haben

Beispiel 2:

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M021	1	Hahn	04509	10
M113	1	Latte	04381	NULL
M117	1	Fahrland	04512	NULL

```
select * from tbl_mitarbeiter
where mitarbeitername like '%a%';
```

Beschreibung:

- alle Mitarbeiter, die an beliebiger Stelle in mitarbeitername den Buchstaben a bzw A haben
- Frage: Welcher Befehl mit einfachem logischen Ausdruck im WHERE würde ebenfalls zu diesem Ergebnis führen?

Beispiel 3:

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M010	1	Stein	04838	12
M021	1	Hahn	04509	10

```
select * from tbl_mitarbeiter
where mitarbeitername like '%n';
```

Beschreibung:

- alle Mitarbeiter, die an der letzten Stelle in mitarbeitername den Buchstaben n bzw N haben

Beispiel 4:

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M021	1	Hahn	04509	10
M117	1	Fahrland	04512	NULL

```
select * from tbl_mitarbeiter
where mitarbeitername like '__h%';
```

Beschreibung:

- alle Mitarbeiter, die an der 3. Stelle in mitarbeitername den Buchstaben h bzw H haben (__ sind in diesem Beispiel 2x _)

4 Verwendung von logischen Operatoren und Ausdrücken im Ausdruck WHERE

Beispiele:

Ausgangstabelle:

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M009	0	Örtel	04105	9
M010	1	Stein	04838	12
M021	1	Hahn	04509	10
M024	0	Holzer	04119	9
M113	1	Latte	04381	NULL
M117	1	Fahrland	04512	NULL
M118	0	Putz	04838	NULL

Ergebnisliste:

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M021	1	Hahn	04509	10
M117	1	Fahrland	04512	NULL

Alle Mitarbeiter mit Maschinenberechtigung aus dem Postleitzahlbereich 045…

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M010	1	Stein	04838	12
M113	1	Latte	04381	NULL

Alle Mitarbeiter mit Maschinenberechtigung, die NICHT aus dem Postleitzahlbereich 045… kommen

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M009	0	Örtel	04105	9
M010	1	Stein	04838	12
M024	0	Holzer	04119	9
M118	0	Putz	04838	NULL

Alle Mitarbeiter mit der Abteilungsnummer 9 oder aus dem Postleitzahlbereich 048…

Syntax:

```
SELECT <spaltenliste> FROM <tabelle>
WHERE <logischer_ausdruck> [<logischer_operator> <logischer_ausdruck>]
```

Beschreibung:

- es werden die Datensätze selektiert, für die der gesamte logische Ausdruck nach dem WHERE den Wahrheitswert TRUE hat
- FALSE wird ausgedrückt durch `0`
- TRUE wird ausgedrückt durch jeden von `0` verschiedenen Wert
- für jeden Datensatz passiert folgendes: für die einzelnen logischen Ausdrücke werden jeweils die Wahrheitswerte ermittelt → alle einzelnen Wahrheitswerte werden über die logische Operatoren verknüpft → der letztendlich übrig bleibende Wahrheitswert entscheidet darüber, ob der Datensatz angezeigt wird oder nicht
- mögliche logische Operatoren sind:
  AND, && → logisches UND
  OR, || → logisches offenes ODER
  NOT, ! → logisches NICHT, Verneinung
  XOR → exclusives logisches ODER
- Quelle: Referenzhandbuch Kapitel 12.1.4

praktische Anwendungen:

Ausgangstabelle:

siehe oben

Ergebnislisten:

Beispiel 1:

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M021	1	Hahn	04509	10
M117	1	Fahrland	04512	NULL

Alle Mitarbeiter mit Maschinenberechtigung aus dem Postleitzahlbereich 045…

```
select * from tbl_mitarbeiter
where maschinenberechtigung = 1 and mitarbeiterPLZ like '045%';
```

Beispiel 2:

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M010	1	Stein	04838	12
M113	1	Latte	04381	NULL

Alle Mitarbeiter mit Maschinenberechtigung, die NICHT aus dem Postleitzahlbereich 045… kommen

```
select * from tbl_mitarbeiter
where maschinenberechtigung = 1 and mitarbeiterPLZ not like '045%';
```

Beispiel 3:

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M009	0	Örtel	04105	9
M010	1	Stein	04838	12
M024	0	Holzer	04119	9
M118	0	Putz	04838	NULL

Alle Mitarbeiter mit der Abteilungsnummer 9 oder aus dem Postleitzahlbereich 048…

```
select * from tbl_mitarbeiter
where abteilungsnummer = 9 or mitarbeiterPLZ like '048%';
```

5 Abfragen mit Listen formulieren

Beispiele:

Ausgangstabelle:

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M009	0	Örtel	04105	9
M010	1	Stein	04838	12
M021	1	Hahn	04509	10
M024	0	Holzer	04119	9
M113	1	Latte	04381	NULL
M117	1	Fahrland	04512	NULL
M118	0	Putz	04838	NULL

Ergebnislisten:

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M010	1	Stein	04838	12
M024	0	Holzer	04119	9

Liste mit allen Mitarbeitern, die Stein oder Holzer heißen

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M009	0	Örtel	04105	9
M010	1	Stein	04838	12
M024	0	Holzer	04119	9

Liste mit allen Mitarbeitern, die in den Abteilungen 9 oder 12 arbeiten

Syntax:

```
SELECT <spaltenliste> FROM <tabelle>
WHERE <spalte> IN (<wert1>, <wert2>[, <wertN>]);
```

Beschreibung:

- <wertN> kann String oder numerischer Wert sein
- die Werte der Liste werden mit einem logischen ODER verknüpft
- der Operator IN kann mit NOT ~~bzw. !~~ negiert werden

praktische Anwendungen:

Ausgangstabelle:

wie oben

Ergebnislisten:

Beispiel 1:

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M010	1	Stein	04838	12
M024	0	Holzer	04119	9

Liste mit allen Mitarbeitern, die Stein oder Holzer heißen

```
select * from tbl_mitarbeiter
where mitarbeitername IN ('Stein', 'Holzer');
```

Beispiel 2:

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M009	0	Örtel	04105	9
M010	1	Stein	04838	12
M024	0	Holzer	04119	9

Liste mit allen Mitarbeitern, die in den Abteilungen 9 oder 12 arbeiten

```
select * from tbl_mitarbeiter
where abteilungsnummer IN (9, 12);
```

6 Abfragen mit Prüfung von Datenfeldern auf NULL bzw. NOT NULL formulieren

Beispiel:

Ausgangstabelle:

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M009	0	Örtel	04105	9
M010	1	Stein	04838	12
M021	1	Hahn	04509	10
M024	0	Holzer	04119	9
M113	1	Latte	04381	NULL
M117	1	Fahrland	04512	NULL
M118	0	Putz	04838	NULL

Ergebnislisten:

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M113	1	Latte	04381	NULL
M117	1	Fahrland	04512	NULL
M118	0	Putz	04838	NULL

Liste der Mitarbeiter, denen keine Abteilungsnummer zugeordnet ist

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M009	0	Örtel	04105	9
M010	1	Stein	04838	12
M021	1	Hahn	04509	10
M024	0	Holzer	04119	9

Liste der Mitarbeiter, denen eine Abteilungsnummer zugeordnet ist

Syntax:

```
SELECT <spaltenliste> FROM <tabelle>
WHERE <spalte> IS [NOT ~~!~~] NULL
```

Erläuterung:

- NULL kann NUR über den Operator IS geprüft werden; mit Gleichheitszeichen = funktioniert das nicht

praktische Anwendungen:

Ausgangstabelle:

wie oben

Ergebnislisten:

Beispiel 1:

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M113	1	Latte	04381	NULL
M117	1	Fahrland	04512	NULL
M118	0	Putz	04838	NULL

Liste der Mitarbeiter, denen **keine Abteilungsnummer** zugeordnet ist

```
select * from tbl_mitarbeiter
where abteilungsnummer is null;
```

Beispiel 2:

Ergebnisliste:

mitarbeiternummer	maschinenberechtigung	mitarbeitername	mitarbeiterPLZ	abteilungsnummer
M009	0	Örtel	04105	9
M010	1	Stein	04838	12
M021	1	Hahn	04509	10
M024	0	Holzer	04119	9

Liste der Mitarbeiter, denen **eine Abteilungsnummer** zugeordnet ist

```
select * from tbl_mitarbeiter
where abteilungsnummer is not null;
```

7 Abfragen mit Bereichen formulieren

Beispiel:

Ausgangstabelle:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
M024	B056	8.00
M010	B021	12.00
M021	B056	21.00
M010	B112	23.00
M021	B112	24.00
M024	B253	24.00
M021	B253	34.00
M009	B253	37.00

(nach stunden_auf_baustelle aufsteigend sortiert)

Ergebnislisten:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
M021	B056	21.00
M010	B112	23.00
M021	B112	24.00
M024	B253	24.00

Liste mit allen Datensätzen, bei denen stunden_auf_baustelle im Bereich von 20 bis 30 liegt

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
M010	B112	23.00
M021	B112	24.00
M024	B253	24.00
M021	B253	34.00

Liste mit allen Datensätzen, bei denen stunden_auf_baustelle im Bereich von 23 bis 34 liegt

Syntax:

```
SELECT <spaltenliste> FROM <tabelle>
WHERE <spalte> BETWEEN <wert1> AND <wert2>
```

Erläuterungen:

- die Grenzwerte <wert1> und <wert2> sind in dem Bereich enthalten
- Grenzwerte <wert1> und <wert2> können numerische Werte oder String sein

Eine alternative Syntax für die Anweisung BETWEEN:

```
SELECT <spaltenliste> FROM <tabelle>
WHERE <spalte> >= <wert1> AND <spalte> <= <wert2>
```

praktische Anwendungen:

Ausgangstabelle:

wie oben

Ergebnislisten:

Beispiel 1:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
M021	B056	21.00
M010	B112	23.00
M021	B112	24.00
M024	B253	24.00

Liste mit allen Datensätzen, bei denen stunden_auf_baustelle im Bereich von 20 bis 30 liegt

```
select * from tbl_ma_auf_baustelle
where stunden_auf_baustelle between 20 and 30
order by stunden_auf_baustelle;
```

Beispiel 2:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
M010	B112	23.00
M021	B112	24.00
M024	B253	24.00
M021	B253	34.00

Liste mit allen Datensätzen, bei denen stunden_auf_baustelle im Bereich von 23 bis 34 liegt

```
select * from tbl_ma_auf_baustelle
where stunden_auf_baustelle between 23 and 34
order by stunden_auf_baustelle;
```

Beispiel 3:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
M010	B021	12.00
M021	B056	21.00
M010	B112	23.00
M021	B112	24.00
M021	B253	34.00

Liste mit allen Datensätzen, bei denen mitarbeiternummer im Bereich von M010 bis M021 liegt

```
select * from tbl_ma_auf_baustelle
where mitarbeiternummer between 'M010' and 'M021'
order by stunden_auf_baustelle;        -- order by ist optional
```

8 Rechnen mit mathematischen Operatoren in Abfragen

Beispiele:

Ausgangstabelle:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
M024	B056	8.00
M010	B021	12.00
M021	B056	21.00
M010	B112	23.00
M021	B112	24.00
M024	B253	24.00
M021	B253	34.00
M009	B253	37.00

(nach stunden_auf_baustelle aufsteigend sortiert)

Ergebnisliste:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle	Stunden + 12	Stunden - 0.75	Stunden * 1.25	Stunden / 3
M009	B253	37.00	49.00	36.25	46.2500	12.333333
M010	B021	12.00	24.00	11.25	15.0000	4.000000
M010	B112	23.00	35.00	22.25	28.7500	7.666667
M021	B056	21.00	33.00	20.25	26.2500	7.000000
M021	B112	24.00	36.00	23.25	30.0000	8.000000
M021	B253	34.00	46.00	33.25	42.5000	11.333333
M024	B056	8.00	20.00	7.25	10.0000	2.666667
M024	B253	24.00	36.00	23.25	30.0000	8.000000

Syntax:

```
SELECT
      <spalte> + <wert1>,       -- Addition
      <spalte> - <wert2>,       -- Subtraktion
      <spalte> * <wert3>,       -- Multiplikation
      <spalte> / <wert4>        -- Division
FROM <tabelle>
```

Beachte:

- Dezimalwerte mit Dezimal-Punkt angeben z.B. 3.1415
- Rangfolge der Operatoren → siehe auch Quelle: Referenzhandbuch Kapitel 12.1.1.

Rangfolge der Operatoren (Auszug):

Priorität	Operator	Beschreibung
HOCH	( )	Klammern
	! NOT	logische Negierung
	+ -	unäres Plus bzw. Minus
	* / %	Division, Modulo
	+ -	Addition, Subtraktion
	=, <=>, >=, >, <=, <, <>, !=, IS, LIKE, IN	Vergleich
	BETWEEN	
	&& AND	logisches UND
	|| OR XOR	logisches ODER, exclusives ODER
NIEDRIG	:=	Wertzuweisung

praktische Anwendung:

Ausgangstabelle:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
M024	B056	8.00
M010	B021	12.00
M021	B056	21.00
M010	B112	23.00
M021	B112	24.00
M024	B253	24.00
M021	B253	34.00
M009	B253	37.00

(nach stunden_auf_baustelle aufsteigend sortiert)

Ergebnisliste:

Beispiel 1:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle	Stunden + 12	Stunden - 0.75	Stunden * 1.25	Stunden / 3
M009	B253	37.00	49.00	36.25	46.2500	12.333333
M010	B021	12.00	24.00	11.25	15.0000	4.000000
M010	B112	23.00	35.00	22.25	28.7500	7.666667
M021	B056	21.00	33.00	20.25	26.2500	7.000000
M021	B112	24.00	36.00	23.25	30.0000	8.000000
M021	B253	34.00	46.00	33.25	42.5000	11.333333
M024	B056	8.00	20.00	7.25	10.0000	2.666667
M024	B253	24.00	36.00	23.25	30.0000	8.000000

```
select
	*,
	stunden_auf_baustelle + 12 'Stunden + 12',
	stunden_auf_baustelle - 0.75 'Stunden - 0.75',
	stunden_auf_baustelle * 1.25 'Stunden * 1.25',
	stunden_auf_baustelle / 3 'Stunden / 3'
from tbl_ma_auf_baustelle;
```

Beispiel 2:

2 * 3 + 4	2 * (3 + 4)	2 + 3 * 4 + 5
10	14	19

```
select
      2 * 3 + 4,
      2 * (3 + 4),
      2 + 3 * 4 + 5;
```

9 Verwendung von Aggregat-Funktionen (Auswahl)

Beispiele:

Ausgangstabelle:

mitarbeiternummer	baustellennummer	stunden_auf_baustelle
M024	B056	8.00
M010	B021	12.00
M021	B056	21.00
M010	B112	23.00
M021	B112	24.00
M024	B253	24.00
M021	B253	34.00
M009	B253	37.00

(nach stunden_auf_baustelle aufsteigend sortiert)

Ergebnislisten:

count(*)
8

Anzahl der Datensätze in der Tabelle

Anzahl der Datensätze in der Tabelle
8

Anzahl der Datensätze in der Tabelle (mit Alias für die Ergebnisspalte)

Anzahl unterschiedlicher Baustellen in der Tabelle
4

Anzahl der Datensätze in der Tabelle mit unterschiedlichen Baustellennummern

Größte Stundenzahl	Kleinste Stundenzahl	Arithmetisches Mittel der Stundenzahlen	Summe der Stundenzahlen
37.00	8.00	22.875000	183.00

Ermittlung verschiedener Werte lt. Spaltenüberschrift aus der Spalte stunden_auf_baustelle

Syntax:

```
SELECT COUNT(* | <spalte>) FROM <tabelle>
```

Erläuterung:

- COUNT(*) → beim Zählen werden alle Spalten der jeweiligen Datensätze berücksichtigt
- COUNT(<spalte>) → beim Zählen werden die Datensätze berücksichtigt, bei denen in der angegebenen Spalte ein Wert ungleich NULL enthalten ist
- Funktion liefert einen einzelnen Wert (sogenannter Skalar oder auch skalarer Wert)

```
SELECT COUNT(DISTINCT <spalte>) FROM <tabelle>
```

Erläuterung:

- beim Zählen werden die Datensätze berücksichtigt, bei denen in der angegebenen Spalte ein Wert ungleich NULL enthalten ist
- weiterhin werden Datensätze, die in der angegebenen Spalte den gleichen Wert haben, jeweils nur 1x gezählt (mehrfaches Zählen des gleichen Werts wird unterdrückt)

```
select
      max(<spalte>) ['<alias_text>'],
      min(<spalte>) ['<alias_text>'],
      avg(<spalte>) ['<alias_text>'],
      sum(<spalte>) ['<alias_text>']
from <tabelle>;
```

Erläuterung:

- MAX() → Ermittlung des größten Werts in der angegebenen Spalte
- MIN() → Ermittlung des kleinsten Werts in der angegebenen Spalte
- AVG() → Ermittlung des arithemitschen Mittelwerts aller Werte in der angegebenen Spalte
- SUM() → Ermittlung der Summe aller Werte in der angegebenen Spalte

praktische Anwendungen:

Ausgangstabelle:
wie oben

Ergebnislisten:

Beispiel 1:

count(*)
8

Anzahl der Datensätze in der Tabelle

```
select count(*) from tbl_ma_auf_baustelle;
```

Beispiel 2:

Anzahl der Datensätze in der Tabelle
8

Anzahl der Datensätze in der Tabelle (mit Alias für die Ergebnisspalte)

```
select count(*) 'Anzahl der Datensätze in der Tabelle'    -- mit Alias
from tbl_ma_auf_baustelle;
```

Beispiel 3:

Anzahl unterschiedlicher Baustellen in der Tabelle
4

Anzahl der Datensätze in der Tabelle mit unterschiedlichen Baustellennummern

```
select count(distinct baustellennummer) 'Anzahl unterschiedlicher Baustellen in der Tabelle'
from tbl_ma_auf_baustelle;
```

Beispiel 4:

Größte Stundenzahl	Kleinste Stundenzahl	Arithmetisches Mittel der Stundenzahlen	Summe der Stundenzahlen
37.00	8.00	22.875000	183.00

Ermittlung verschiedener Werte lt. Spaltenüberschrift aus der Spalte stunden_auf_baustelle

```
select
       max(stunden_auf_baustelle) 'Größte Stundenzahl',
       min(stunden_auf_baustelle) 'Kleinste Stundenzahl',
       avg(stunden_auf_baustelle) 'Arithmetisches Mittel der Stundenzahlen',
       sum(stunden_auf_baustelle) 'Summe der Stundenzahlen'
from tbl_ma_auf_baustelle;
```