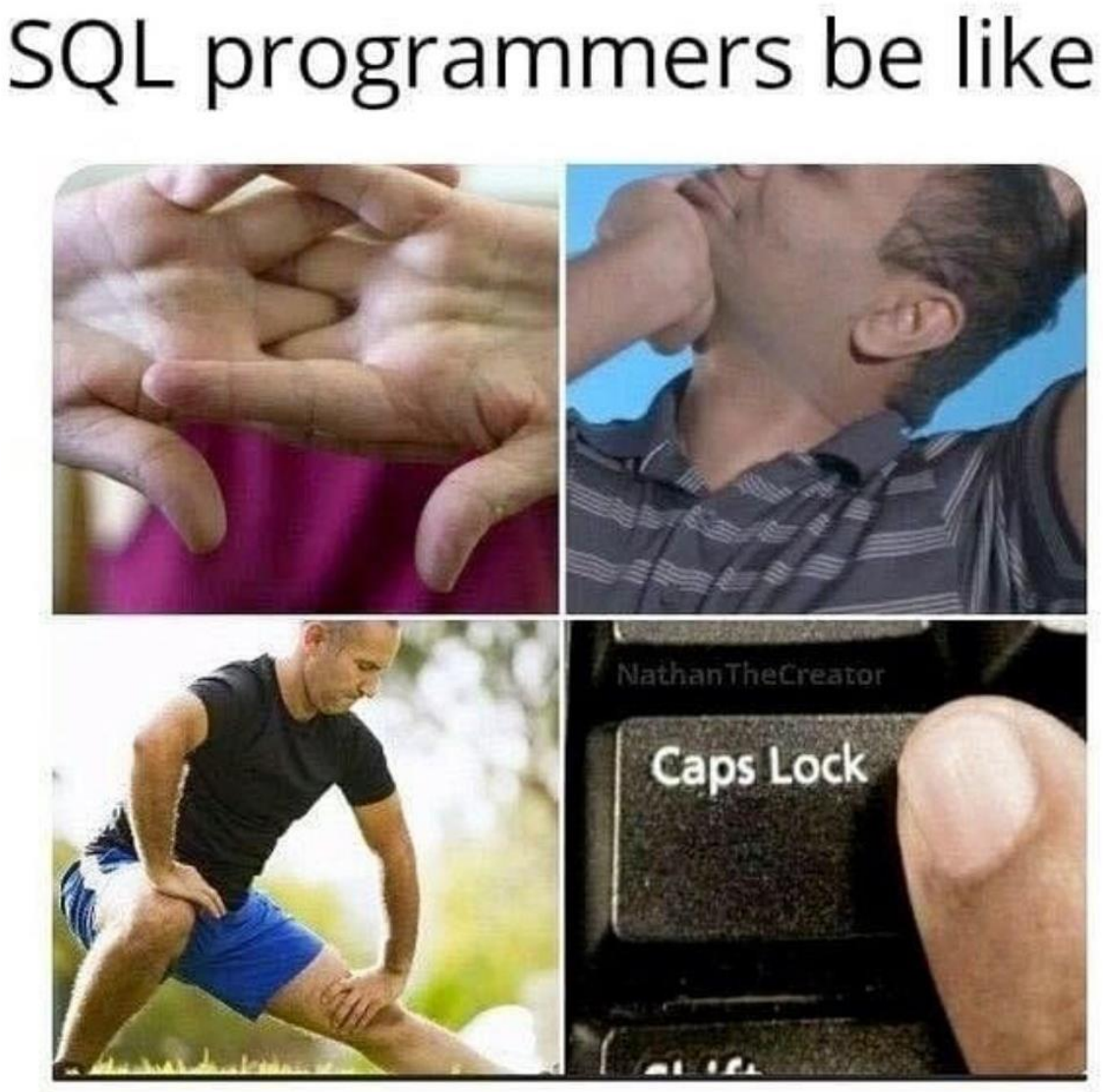
Tutorium 5 – Datenbanksysteme

30.11.2022 – Finn Kapitza



LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

1. Wiederholung
SQL – Anfragen

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

SQL – Anfragen

- SQL ist *case insensitive* -> Groß-/Kleinschreibung bei den **Befehlen** nicht wichtig, bei Relationen-/Attributwerten und Konstanten aber schon

- Immer nur die Befehle verwenden die auch in der Vorlesung vorgestellt wurden

- Schnittstelle zum Ausprobieren/Üben auf Uni2Work benutzen
  - Wenn man nicht im *eduroam* ist dann muss man eduVPN verwenden
  - Link zur Schnittstelle: http://sql-demo.dbs.ifi.lmu.de/
  - Link zu eduVPN: https://doku.lrz.de/display/PUBLIC/VPN+-+eduVPN+-+Installation+und+Konfiguration

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

SQL - Anfragen

- SQL Anfragen bestehen immer aus:

1. **SELECT** …
2. **FROM**…

- und meistens auch aus:

3. **WHERE** … oder GROUP BY … HAVING … (kommt erst später)

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

SELECT Klausel

- **SELECT** wird zum Auswählen (Projizieren) von Attributen/Spalten verwendet
- Kommt immer als erstes

Optionen:

1. **SELECT** * -> Anzeigen aller Attribute/Spalten

2. **SELECT** attr_1, …, attr_n -> Anzeigen der angegebenen Attribute/Spalten

3. **SELECT DISTINCT** attr_1, …, attr_n -> wie 2. nur mit Duplikat Eliminierung

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

FROM Klausel

- **FROM** wird zum Auswählen von Relationen und Ausführen von Joins verwendet
- Kommt immer nach **SELECT**

Optionen:

1. Relation_1**,** …**,** Relation_n oder Relation_1 **CROSS JOIN** Relation_n
   -> Kreuprodukt der angegebenen Relationen

2. Relation_1 **NATURAL JOIN** Relation_2 … **NATURAL JOIN** Relation_n
   -> Natural Join (alle gleichnamigen Attribute) der angegebenen Relationen
   -> Nur eine Kopie der gleichnamigen Attribute bleibt
   -> Wenn keine gleichnamigen Attribute, dann Kreuzprodukt

FROM Klausel

3. Relation_1 **JOIN** Relation_2 **ON** Relation_1.attribut_i **Θ** Relation_2.attribut_j
   -> Theta Join mit $\Theta \in \{=, \leq, <, >, \geq, \neq\}$
   -> Wenn $\Theta \in \{=\}$, dann Equi Join (Attribute werden nicht eliminiert)

4. Relation_1 **JOIN** Relation_2 **USING** (attribut_i)
   -> attribut_i muss bei beiden Relationen gleich heißen

- Es können auch immer mehrere Joins kombiniert werden
  -> Dann aber immer Klammern um die Joins setzen

WHERE Klausel

- In **WHERE** kann man Bedingungen an Tupel aus der aus **FROM** resultierenden Relation stellen (Selektion)

Optionen (Beispiele):

- Relation_1.attr_i Θ Relation_2.attr_j mit $\Theta \in \{=, \leq, <, >, \geq, \neq\}$
  -> Gleicher Datentyp von attr_i und attr_j
- Attr_j = 5
- Attr_k = „Hallo“

=> Verknüpfung mehrerer Bedingungen mit **AND** und **OR** möglich (Achte auf Klammersetzung)

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Aufgabe 5.1

- Anfragen immer formulieren in:

1. Relationaler Algebra
2. SQL

DISTINCT soll nur verwendet werden, falls es notwendig ist

Aufgabe 5.1.a – Bestimmen Sie die Artikelnummern und Bezeichnungen des gesamten Inventars mit Lagerort in Koeln und einem Preis von mindestens 4500

1. Relationale Algebra:

- Alle Informationen stehen in der Relation *Inventar*
- Projektion auf: *art_nr* und *art_bez*
- Selektion auf: *lagerort* und *preis*

$$\Pi_{art\_nr,art\_bez}(\sigma_{lagerort=´Koeln` \wedge preis \geq 4500}(Inventar))$$

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Aufgabe 5.1.a – Bestimmen Sie die Artikelnummern und Bezeichnungen des gesamten Inventars mit Lagerort in Koeln und einem Preis von mindestens 4500

2. SQL:

- Alle Informationen stehen in der Relation *Inventar*
- **SELECT**: *art_nr* und *art_bez*
- **WHERE**: *lagerort* und *preis*

SELECT art_nr, art_bez

FROM Inventar

WHERE lagerort=“Koeln” **AND** preis >= 4500**;**

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Aufgabe 5.1.b – Bestimmen Sie für jeden Auftragsposten die Auftragsnummer, die Nummer und Bezeichnung des Artikels und dessen im Auftrag geforderte Menge

1. Relationale Algebra:

- Auftragsnummer, Artikelnummer und Menge stehen in *Auftragsposten*
- Artikelbezeichnung steht in Relation *Inventar*
- Projektion auf: *auftr_nr, art_nr, art_bez* und *menge*

$$\Pi_{auftr\_nr,art\_nr,art\_bez,menge}(Inventar \bowtie Auftragsposten)$$

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Aufgabe 5.1.b – Bestimmen Sie für jeden Auftragsposten die Auftragsnummer, die Nummer und Bezeichnung des Artikels und dessen im Auftrag geforderte Menge

2. SQL:

- Relationen *Inventar* und *Auftragsposten* -> nur **NATURAL JOIN**
- **SELECT**: Auftragsnummer, Artikelnummer, Artikelbezeichnung und Menge
- **DISTINCT**: Artikel mit gleicher art_nr können an verschiedenen lagerorten gelagert sein

SELECT DISTINCT auftr_nr, art_nr, art_bez, menge

FROM Inventar **NATURAL JOIN** Auftragsposten**;**

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Aufgabe 5.1.c – Finden Sie die Bezeichnungen aller Artikel, die in einem Ort Lagern in dene auch Kunden ihren Sitz/Wohort haben und die teurer als 7000 sind

1. Relationale Algebra:

- Preis und Lagerort stehen in Relation *Inventar*
- Wohnort steht in Relation *Kunde*
- Projektion auf: *art_nr*
- Seletion auf: preis > 7000 und ort = lagerort **ODER** Theta-Join

$$\Pi_{art\_nr}(\sigma_{preis>7000 \wedge ort=lagerort}(Inventar \bowtie Kunde)$$

ODER

$$\Pi_{art\_nr}(\sigma_{preis>7000}(Inventar \underset{ort=lagerort}{\bowtie} Kunde)$$

Aufgabe 5.1.c – Finden Sie die Bezeichnungen aller Artikel, die in einem Ort Lagern in dene auch Kunden ihren Sitz/Wohort haben und die teurer als 7000 sind

2. SQL:

- Relationen *Inventar* und *Kunde* -> nur **JOIN ON** Konstrukt
- **SELECT**: Artikelbezeichnung
- **WHERE**: lagerort gleich ort und preis

SELECT DISTINCT art_bez

FROM Inventar I **JOIN** Kunde K **ON** (I.lagerort=K.ort)

WHERE I.preis > 7000**;**

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Aufgabe 5.1.d – Finden Sie für jeden Kunden die Kundennummer sowie die Nummern und Bezeichnungen aller von ihm gekauften Artikel, die im selben Ort lagern, in dem er der Sitz/Wohnort hat und die günstiger als 7000 sind

1. Relationale Algebra:

- Preis und Lagerort stehen in Relation *Inventar*
- Wohnort und Kundennummer steht in Relation *Kunde*
- Verbunden durch Auftragsposten und Verkauf
- Projektion auf: *kund_nr, art_nr* und *art_bez*
- Seletion auf: ort = lagerort und preis < 7000

$$\Pi_{kund\_nr,art\_nr,art\_bez}(\sigma_{preis<7000 \wedge ort=lagerort} (Kunde \bowtie (Verkauf \bowtie (Auftragsposten \bowtie Inventar))))$$

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Aufgabe 5.1.d – Finden Sie für jeden Kunden die Kundennummer sowie die Nummern und Bezeichnungen aller von ihm gekauften Artikel, die im selben Ort lagern, in dem er der Sitz/Wohnort hat und die günstiger als 7000 sind

2. SQL:

- Relationen *Inventar, Kunde, Auftragsposten* und *Verkauf* -> **JOIN USING**
- **SELECT**: Kundennummer, Artikelnummer und Artikelbezeichnung
- **WHERE**: lagerort gleich ort und preis

SELECT DISTINCT kund_nr, art_nr, art_bez

FROM Verkauf **JOIN** Kunde **USING** (kund_nr) **JOIN** Auftragsposten **USING** (auftr_nr) **JOIN** Inventar **USING** (art_nr)

WHERE lagerort=ort **AND** preis < 7000**;**

Aufgabe 5.1.e – Bestimmen Sie die Nummern und Nachnamen aller Mitarbeiter, die sowohl den Artikel mit der Nummer 104003 als auch den Artikel mit der Nummer 401001 verkauft haben

1. Relationale Algebra:

- Pers_nr und nachname steht in Relation *Personal*
- Art_nr von einem Verkauf steht in Relation *Auftragsposten*
- Relation *Verkauf* wird noch zum Verbinden benötigt
- Projektion auf Pers_nr und nachname
- Selektion auf art_nr = 104003 oder art_nr = 401001

$$\Pi_{pers\_nr,nachname}(\sigma_{art\_nr=104003}(Personal \bowtie (Verkauf \bowtie Auftragsposten)$$
$$\cap$$
$$\Pi_{pers\_nr,nachname}(\sigma_{art\_nr=401001}(Personal \bowtie (Verkauf \bowtie Auftragsposten)$$

Aufgabe 5.1.e – Bestimmen Sie die Nummern und Nachnamen aller Mitarbeiter, die sowohl den Artikel mit der Nummer 104003 als auch den Artikel mit der Nummer 401001 verkauft haben

2. SQL:

- Relationen *Personal, Auftragsposten* und *Verkauf* -> **CROSS JOIN**
- **SELECT**: Personalnummer, Mitarbeiternachname
- **Bedingung**: artikel mit Nummer 104003 und 401001 verkauft

- **INTERSECT** gibt es auch in SQL
- **SELF JOIN** in SQL möglich
- Anfrage auch mit **IN** möglich

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Aufgabe 5.1.e – Intersect

```
SELECT DISTINCT P.pers_nr, p.nachname

FROM Personal P, Verkauf V, Auftragsposten A

WHERE P.pers_nr = V.pers_nr AND V.auftr_nr = A.auftr_nr AND A.art_nr=104003

INTERSECT

SELECT DISTINCT P.pers_nr, p.nachname

FROM Personal P, Verkauf V, Auftragsposten A

WHERE P.pers_nr = V.pers_nr AND V.auftr_nr = A.auftr_nr AND A.art_nr=401001
```

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Aufgabe 5.1.e – Self Join

SELECT DISTINCT P.pers_nr, P.nachname

FROM Personal P**,** Verkauf V1**,** Verkauf V2**,** Auftragsposten A1**,** Auftragsposten A2

WHERE P.pers_nr = V1.pers_nr **AND** P.pers_nr = V2.pers_nr **AND**

V1.auftr_nr = A1.auftr_nr **AND** V2.auftr_nr = A2.auftr_nr **AND**

A1.art_nr = 104003 **AND** A2.art_nr = 401001

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

2. Wiederholung
SQL – Änderungsoperationen

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

SQL Änderungsoperationen

Werden benutzt um:

- Neue Tupel in Relationen einzufügen
- Bestehende Tupel zu ändern/aktualisieren
- Tupel aus der Relation zu löschen

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Änderungsoperationen – Einfügen von Tupeln

INSERT INTO Relation [(attr_1, …, attr_n)]
VALUES (value_1, …, value_n), (value_2, …, value_m)

- Erlaubt einfügen von mehreren Tupeln auf einmal
- Erlaubt einfügen von nicht vollständigen Tupeln

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Änderungsoperationen – Aktualisieren von Tupeln

UPDATE Relation
SET attr_1=wert_1 [, attr_2=wert_2 …]
[**WHERE** Bedingung]

- Erlaubt das aktualisieren von bestimmten Tupeln (Bedingung)
- Bedingung ist optional, wird aber fast immer verwendet
- Bedingung in **WHERE** ist gleich zu den Anfragen

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Änderungsoperationen – Löschen von Tupeln

DELETE FROM Relation
[**WHERE** Bedingung]

- Löscht alle Tupel, die Bedingung erfüllen

- Bedinung ist optional wird aber eigentlich immer verwendet

Aufgabe 5.2 – Änderungsoperationen in SQL

- Die Fremdschlüssel wurden mit *on delete cascade* definiert
  -> Tupel die auf Primärschlüssel referenzieren werden automatisch gelöscht wenn der referenzierte Primärschlüssel gelöscht wird

- Manchmal ist die Reihenfolge der Operationen wichtig

- Datenbank entspricht der bekannten Beispielausprägung

Aufgabe 5.2.a – Es werden 12 Möbelstücke mit der Bezeichnung „FATBOY SITZSACK“ an das Lager in Hamburg geliefert. Artikelnummer ist ‚424242‘ und Preis ist ‚90.00‘

INSERT INTO Inventar (art_nr, art_bez, lagerbest, lagerort, preis)

VALUES (424242, “FATBOY SITZSACK”, 12, “Hamburg”, 90.00)

Oder

INSERT INTO Inventar

VALUES (424242, “FATBOY SITZSACK”, 12, “Hamburg”, 90.00)

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Aufgabe 5.2.b – Die Manager des Unternehmens (d.h. Personen mit vorgesetzt=0) gönnen sich eine Gehaltserhöhung. Verdoppeln Sie deren Gehalt.

```
UPDATE Personal

SET gehalt = 2 * gehalt

WHERE vorgesetzt = 0;
```

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Aufgabe 5.2.c

1. Der Manager Sander Ernst (pers_nr = 8) wird in die Frührente entlassen. Seine Personalien warden aus der Datenbank gelöscht.

2. Gabriele Neumann (pers_nr = 12) wird nach München versetzt. Ihr Gehalt um 1000 erhöht und Sie wird zu Managerin befördert (vorgesetzt = 0)

3. Frau Neumann übernimmt die Betreuung aller Untergebenen und Verkäufe von Herrn Ernst.

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Aufgabe 5.2.c

- Als Erstes Befördern wir Frau Neumann

UPDATE Personal

SET einsatz=„Muenchen“, gehalt = gehalt + 1000, vorgesetzt = 0

WHERE pers_nr = 12

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Aufgabe 5.2.c

- Dann Aktualisieren wir alle Tupel der Untergebenen von Herrn Ernst und der Verkäufe von Herrn Ernst

UPDATE Personal

SET vorgesetzt = 12

WHERE vorgesetzt = 8

UPDATE Verkauf

SET pers_nr = 12

WHERE pers_nr = 8

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Aufgabe 5.2.c

- Dann löschen wir Herrn Ernst‘ Personalien aus der Datenbank

DELETE FROM Personal

WHERE pers_nr = 8

Aufgabe 5.3 – Join-Operationen in SQL

Im Folgenden markiert das Symbol $\rtimes$ einen Left-Outer-Join und das Symbol $\ltimes$ eine Right-Outer-Join.

Beide Symbole können zur Modifikation von Theta-, Equi- oder Natural-Join genutzt werden.

So stellt z.B $\underset{T1.nr=T2.nr}{\rtimes}$ den Left-Outer-Equi-Join da.

Gegeben sind die beiden Relationen:

T1

nr	name
1	a
4	b
5	c

T2

nr	wert
4	w
5	x
5	y
7	z

Aufgabe 5.3.a - $T1 \times T2$

SELECT *

FROM T1 **CROSS JOIN** T2;

T1

nr	name
1	a
4	b
5	c

T2

nr	wert
4	w
5	x
5	y
7	z

nr	name	nr	wert
1	a	4	w
1	a	5	x
1	a	5	y
1	a	7	z
4	b	4	w
4	b	5	x
4	b	5	y
4	b	7	z
5	c	4	w
5	c	5	x
5	c	5	y
5	c	7	z

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Aufgabe 5.3.b - $T \underset{T1.nr=T2.nr}{\bowtie} T2$

SELECT *

FROM T1 **JOIN** T2 **ON** T1.nr = T2.nr;

T1

nr	name
1	a
4	b
5	c

T2

nr	wert
4	w
5	x
5	y
7	z

nr	name	nr	wert
4	b	4	w
5	c	5	x
5	c	5	y

Aufgabe 5.3.b - $T1 \bowtie T2$

```
SELECT *

FROM T1 NATURAL JOIN T2;
```

T1

nr	name
1	a
4	b
5	c

T2

nr	wert
4	w
5	x
5	y
7	z

nr	name	wert
4	b	w
5	c	x
5	c	y

Aufgabe 5.3.b – T1 $\underset{T1.nr=T2.nr}{\ltimes}$ T2

SELECT *

FROM T1 **RIGHT JOIN** T2 **ON** T1.nr = T2.nr;

T1

nr	name
1	a
4	b
5	c

T2

nr	wert
4	w
5	x
5	y
7	z

nr	name	nr	wert
4	b	4	w
5	c	5	x
5	c	5	y
		7	z

Aufgabe 5.3.b – T1 $\underset{T1.nr=T2.nr}{\bowtie} T2$

SELECT *

FROM T1 **LEFT JOIN** T2 **ON** T1.nr = T2.nr;

T1

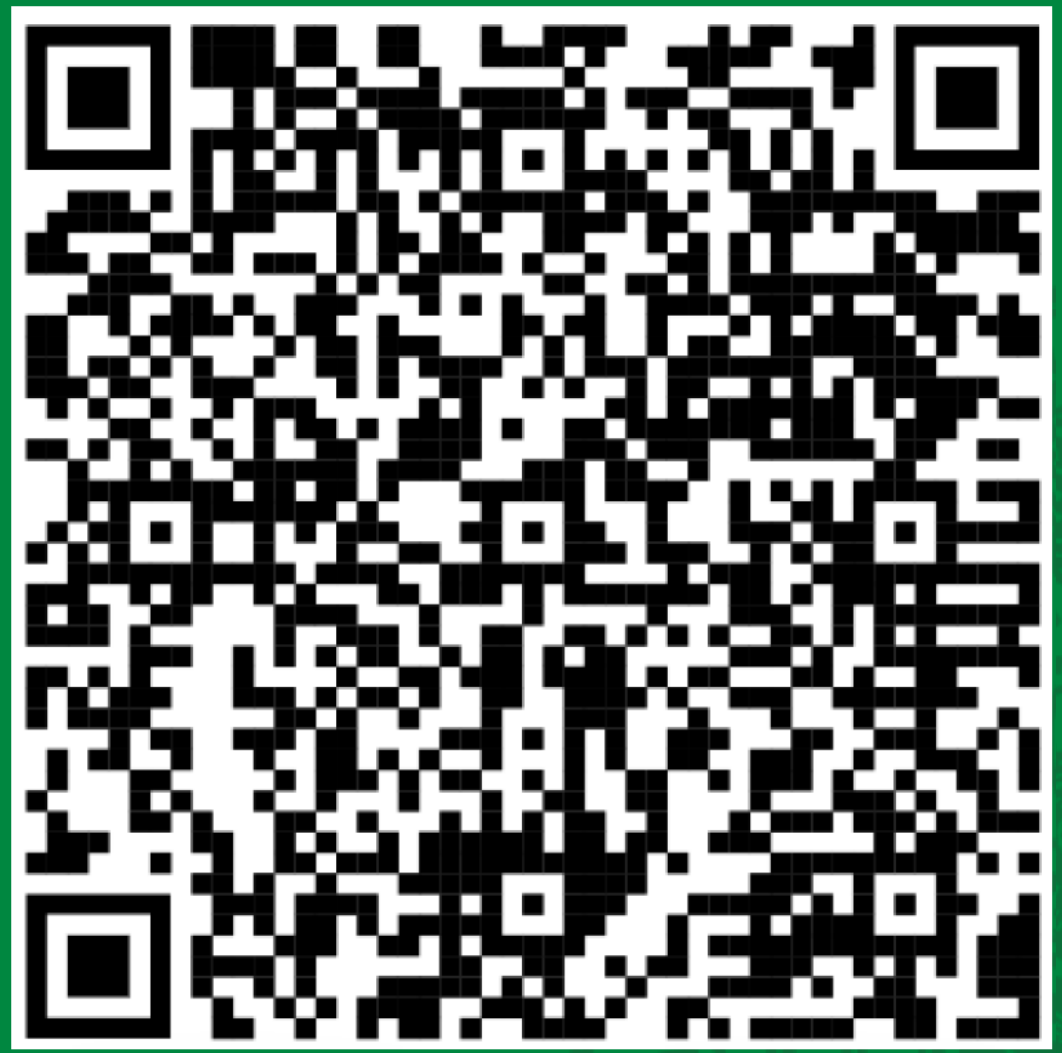
nr	name
1	a
4	b
5	c

T2

nr	wert
4	w
5	x
5	y
7	z

nr	name	nr	wert
1	a		
4	b	4	w
5	c	5	x
5	c	5	y

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN



Finn Kapitza – Finn.Kapitza@campus.lmu.de