Gruppe **A**

Bitte tragen Sie **SOFORT** und **LESERLICH** Namen und Matrikelnr. ein, und legen Sie Ihren Studentenausweis bereit.

PRÜFUNG AUS	**MUSTERLÖSUNG**	29.01.2019
◯ DATENMODELLIERUNG 2 (**184.790**)	◯ DATENBANKSYSTEME (**184.686**)	**GRUPPE A**

Matrikelnr.	Familienname	Vorname

Arbeitszeit: 90 Minuten. Lösen Sie die Aufgaben auf den vorgesehenen Blättern; Lösungen auf Zusatzblättern werden nicht gewertet. **Viel Erfolg!**

Aufgabe 1: Eigenschaften von Transaktionen (11)

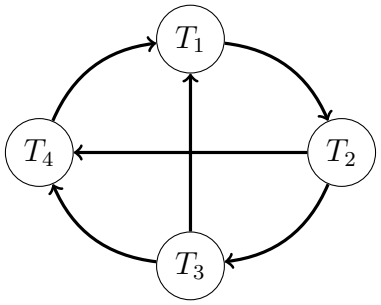
Angabe zu Aufgabe 1a)

T_1: $b_1, r_1(A), r_1(A), w_1(C), c_1$

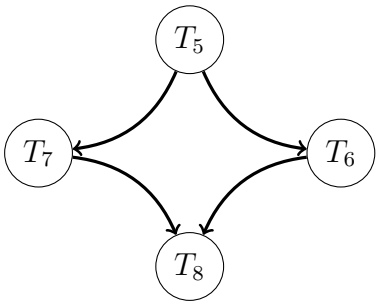
T_2: $b_2, r_2(B), w_2(A), c_2$

T_3: $b_3, r_3(C), r_3(A), c_3$

T_4: $b_4, r_4(C), w_4(C), r_4(A), c_4$



Angabe zu Aufgabe 1b)



a) Gegeben sind die Elementaroperationen von vier Transaktionen T_1, T_2, T_3, T_4 welche auf den drei Datensätzen A, B und C arbeiten. Dabei bezeichnet $r_i(O)$ eine Lese- und $w_i(O)$ eine Schreiboperation von Transaktion T_i, sowie b_i und c_i den Beginn bzw. das Commit einer Transaktion. Gegeben ist außerdem ein Serialisierbarkeitsgraph.

Geben Sie eine Historie dieser vier Transaktionen an (also eine möglicherweise verzahnte Ausführung welche die Reihenfolge der Aktionen innerhalb der einzelnen Transaktionen nicht ändert), welche genau den angebenen Serialisierbarkeitsgraphen erzeugt. Achten Sie insbesonders darauf, dass Ihre Historie keine weiteren Kanten im Serialisierbarkeitsgraphen erzeugt, selbst wenn sich diese transitiv aus bestehenden Kanten ableiten lassen.

$$b_1, b_2, b_3, b_4, r_3(C), r_4(C), w_4(C), r_1(A), r_2(B), r_1(A), w_2(A), w_1(C), r_3(A), r_4(A), c_1, c_2, c_3, c_4$$

b) Gegeben ist der oben rechts dargestellte Serialisierbarkeitsgraph. Geben Sie eine (möglichst kurze) Historie (bestehend aus Lese- bzw. Schreiboperationen $r_i(O)$ und $w_i(O)$) der vier Transaktionen T_5, T_6, T_7 und T_8 über den Datensätzen P und Q (es sind keine weiteren Datensätze erlaubt) an, welche *exakt* diesen Serialiesierbarkeitsgraph erzeugt. (Achten Sie insbesonders darauf dass Ihre Historie keine Kante $T_5 \rightarrow T_8$ im Serialisierbarkeitsgraph erzeugt.)

$$b_5, b_6, b_7, b_8, r_5(P), r_5(Q), w_6(P), w_7(Q), r_8(P), r_8(Q), c_5, c_6, c_7, c_8$$

c) Unabhängig davon, ob Ihre Lösung bei a) richtig ist oder nicht: Ist eine korrekte Lösung für a) *konfliktserialisierbar*? Begründen Sie kurz (**1 Satz**) Ihre Antwort. Falls ja, geben Sie bitte eine konfliktäquivalente, serielle Reihenfolge der Transaktionen an. Falls nicht, geben Sie eine Transaktion an die man abbrechen muss, damit die verbliebenen Transaktionen konfliktserialisierbar sind.

Konfliktserialisierbar: ◯ ja ⊗ nein

Begründung: `Der Serialisierbarkeitsgraph ist nicht azyklisch. Nach dem .....`

`Abbruch von` T_1 `sind die verbleibenden Transaktionen konfliktserialisierbar ....`

(*Achtung:* Ankreuzen alleine gibt keine Punkte!)

Aufgabe 2: Protokollierung (Logging) (13)

Diese Aufgabe behandelt die Protokollierung und Recovery nach dem ARIES Verfahren. Dazu verwenden wir die selbe Notation wie in der Vorlesung, welche im folgenden beschrieben ist:

Wir verwenden das Format `[LSN, TA, PageID, Redo, Undo, PrevLSN]` *für "normale" Logeinträge, und das Format* ⟨`LSN, TA, PageID, Redo, PrevLSN, UndoNextLSN`⟩ *für Kompensations Logeinträge (Compensation Log Records). Für* `BOT` *und* `COMMIT` *Log-Einträge kann das Format* `[LSN, TA, BOT, PrevLSN]` *bzw.* `[LSN, TA, COMMIT, PrevLSN]` *verwendet werden. Nehmen Sie zur Vereinfachung an, dass die Felder A, B und C jeweils auf den Seiten P_A, P_B bzw. P_C liegen.*

*Beachten Sie, dass Undo/Redo-Einträge **relativ** zum Datenbestand mittels **Addition** bzw. **Subtraktion** angegeben werden, z.B.: [#i, T_j, P_X, X+=d_1, X-=d_2, #k] bedeutet, dass laut i-tem Logeintrag die Transaktion T_j auf ein Datum X auf der Seite P_X schreibend zugreift, so dass beim Redo X um d_1 vergrößert werden müsste und beim Undo X um d_2 verkleinert werden müsste. Außerdem hat der vorangegangene Logeintrag dieser Transaktion die Nummer k.*

$r(X, y_i)$ gibt an, dass der Wert des Feldes X in die lokale Variable y_i gelesen wird. $w(X, y)$ gibt an, dass der Wert y in das Feld X geschrieben wird.

Historie für Aufgaben 2a) und 2b)

Schritt	T_1	T_2	T_3
1			`BOT`
2		`BOT`	
3	`BOT`		
4			$w(C, 50)$
5			$r(B, b_3)$
6	$r(A, a_1)$		
7	$r(B, b_1)$		
8		$r(C, c_2)$	
9	$w(B, a_1 + b_1)$		
10			$r(A, a_3)$
11		$w(C, c_2 + 5)$	
12			$w(B, b_3 - 10)$
13	$w(A, a_1 - 5)$		
14		`COMMIT`	
15			$r(C, c_3)$
16	$w(C, 50)$		

Logeinträge für Aufgabe 2c)

[#1, T_1, `BOT`, #0]
[#2, T_1, P_A, A+=10, A-=10, #1]
[#3, T_3, `BOT`, #0]
[#4, T_3, P_C, C+=5, C-=5, #3]
[#5, T_2, `BOT`, #0]
[#6, T_2, P_B, B+=35, B-=35, #5]
[#7, T_3, P_C, C+=10, C-=10, #4]
[#8, T_2, P_C, C-=20, C+=20, #6]
[#9, T_3, P_A, A+=40, A-=40, #7]
[#10, T_3, P_B, B-=37, B+=37, #9]
⟨#11, T_3, P_B, B+=37, #10, #9⟩
[#12, T_2, `COMMIT`, #8]
[#13, T_1, P_A, A+=1, A-=1, #2]
⟨#14, T_3, P_A, A-=40, #11, #7⟩
[#15, T_1, P_C, C+=3, C-=3, #13]

a) Gegeben ist (auf der vorigen Seite) die Historie dreier Transaktionen T_1, T_2 und T_3. Nehmen Sie an, dass zu Beginn der Historie der relevante Datenbestand der Datenbank aus folgenden Werten besteht:

$$A = 25,\ B = 40 \text{ und } C = 30$$

Geben Sie an, welche Log-Einträge beim Abarbeiten der Historie erstellt werden. Verwenden Sie dazu das zuvor beschriebene Format. Die *begin of transaction* Einträge sind bereits vorgegeben.

`[#1, T3, BOT, #0], [#2, T2, BOT, #0], [#3, T1, BOT, #0]`

`[#4, T3, PC, C+=20, C-=20, #1]`	`[#8, T1, PA, A-=5, A+=5, #5]` .....
`[#5, T1, PB, B+=25, B-=25, #3]`	`[#9, T2, COMMIT, #6]` .....
`[#6, T2, PC, C+=5, C-=5, #2]`	`[#10, T1, PC, C-=5, C+=5, #8]` .....
`[#7, T3, PB, B-=35, B+=35, #4]`	.....

b) Kommt es in der Historie von Aufgabe a) zu einem *Lost Update*? Falls ja, geben Sie bitte an welches Update verloren geht, und warum. Falls nicht, geben Sie an wie man die Historie verändern oder erweitern könnte, so dass es zu einem Lost Update kommt.

Es kommt zu einem Lost Update: ⊗ ja ○ nein

Begründung: Das Update $w(B, a_1 + b_1)$ von T_1 geht verloren: Transaktion T_3 .....
liest den Wert von B in Zeile 5 (vor dem Update) und überschreibt dann das ..
Update von T_1, ohne dass dieses Update dabei berücksichtigt würde. Das .....
Lost Update ergibt sich daher aus $r_3(B, b_3)$, $r_1(B, b_1)$, $w_1(B, a_1 + b_1)$, $w_3(B, b_3 - 10)$

(*Achtung:* Ankreuzen alleine gibt keine Punkte!)

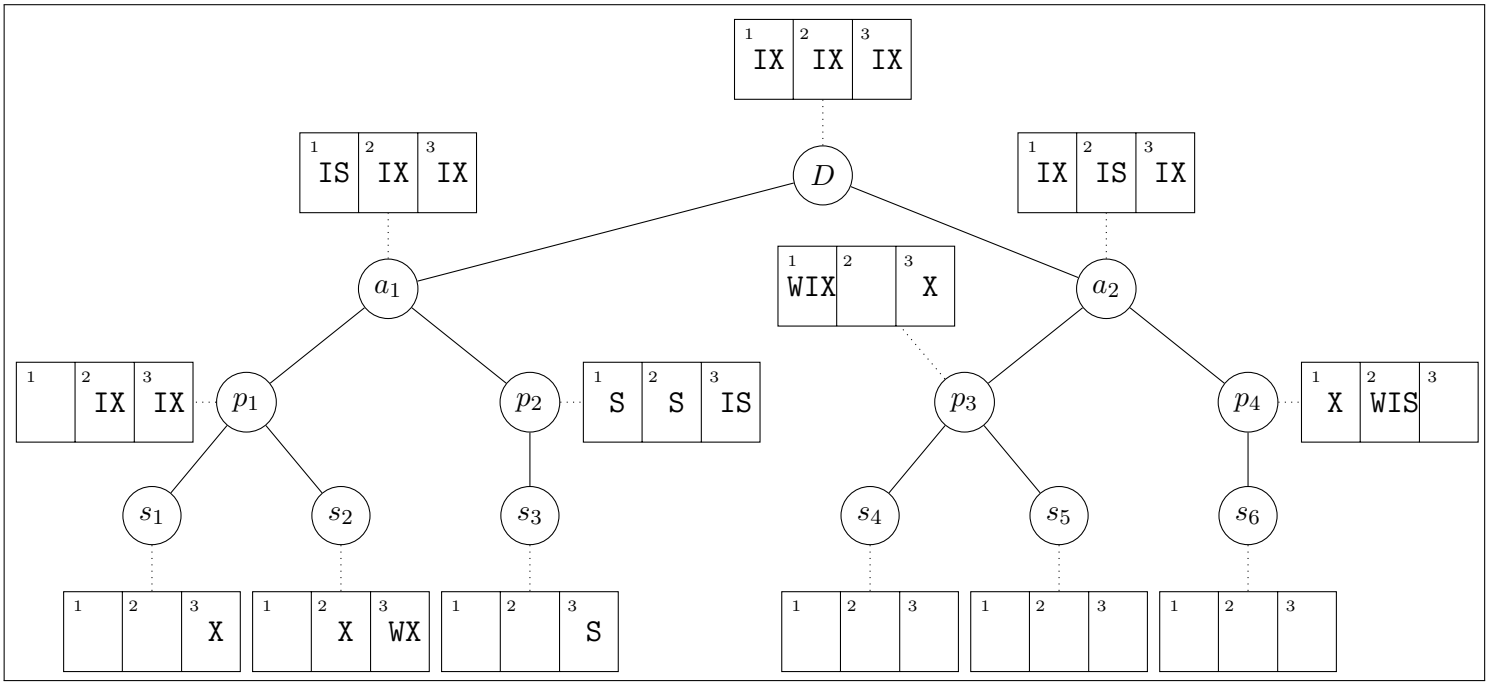
c) Betrachten Sie die auf der vorigen Seite angebenen Log-Einträge, und nehmen Sie an, dass an Hand dieser Log-Einträge ein Wiederanlauf (Recovery) durchgeführt wird. Geben Sie an, welche Log-Einträge dabei erstellt werden. Verwenden Sie dazu das zuvor beschriebene Format.

⟨#16, T_1, P_C, C-=3, #15, #13⟩	⟨#20, T_3, BOT, #19⟩ .....
⟨#17, T_1, P_A, A-=1, #16, #2⟩	⟨#21, T_1, P_A, A-=10, #17, #1⟩ .....
⟨#18, T_3, P_C, C-=10, #14, #4⟩	⟨#22, T_1, BOT, #21⟩ .....
⟨#19, T_3, P_C, C-=5, #18, #3⟩	.....
.....	.....

Aufgabe 3: Mehrbenutzersynchronisation (11)

a) Multi Granularity Locking:

Gegeben ist folgende Datenbasis-Hierarchie.



Nehmen Sie an, die Transaktionen T_1, T_2 und T_3 möchten folgenden Sperren in der angegeben Reihenfolge erhalten

$$XL_2(s_2), XL_3(s_1), XL_3(p_3), SL_2(p_2), XL_1(p_4), SL_2(s_6), SL_1(p_2), SL_3(s_3), XL_1(s_5), XL_3(s_2), XL_2(s_3), SL_2(p_3)$$

Dabei bezeichnen $SL_i(O)$ und $XL_i(O)$ den Wunsch von Transaktion T_i eine Lese- bzw. Schreibsperre auf das Objekt O zu erhalten. Nehmen Sie weiter an, dass die Transaktionen zum Erhalt dieser Sperren nach dem Sperrprotokoll des MGL vorgehen (und dieses auch korrekt einhalten; beachten Sie, dass obige Angabe nur die Sperren beinhaltet, welche die Transaktionen haben wollen, und dass zum Erhalt dieser Sperren zusätzliche Sperren nötig sind).

Beschreiben Sie den Zustand nach Abarbeitung dieser Sperranforderungen, indem Sie in der obigen Grafik zu jedem Knoten notieren, welche Transaktionen welche Sperren auf diesem Knoten halten. Tragen Sie dazu S, X, IS und IX in das Feld mit der Transaktionsnummer ein um auszudrücken, dass die Transaktion die entsprechende Sperre hält. Sollte eine Transaktion eine Sperre angefordert aber nicht erhalten haben, tragen Sie bitte WS, WX, WIS oder WIX ein. Sollte eine Transaktion blockieren, ignorieren Sie alle weiteren Anforderungen dieser Transaktion.

b) Deadlocks:

Besteht nach Abarbeitung der Anforderungen in Aufgabe a) ein Deadlock? Falls ja, eklären Sie kurz warum (geben Sie an, welche Transaktion warum auf welche andere Transaktion wartet). Falls nein, geben Sie eine (möglichst kurze) Folge von Sperranforderungen an, welche man nach den Anforderungen aus Aufgabe a) ausführen müsste um einen Deadlock zu erhalten.

Es besteht ein Deadlock: ⊗ ja ○ nein

Begründung: T_1 wartet in p_3 auf T_3, ...

welche wiederum in s_3 auf T_2 wartet, ...

welche wiederum in p_4 auf T_1 wartet. .....

(*Achtung:* Ankreuzen alleine gibt keine Punkte!)

c) Weitere Sperrprotokolle:

Gegeben ist die untenstehende Folge von Sperranforderungen, Freigaben von Sperren, sowie Lese- und Schreiboperationen. Die Bedeutung von $XL_i(O)$ und $SL_i(O)$ ist wie zuvor, $relSL_i(O)$ (bzw. $relXL_i(O)$) bedeuten dass die Transaktion T_i eine gehaltene Lesesperre (bzw. eine Schreibsperre) auf das Datenobjekt O freigibt, und $r_i(O)$, $w_i(O)$ und c_i sind wie in Aufgabe 1.

Welche der vier Transaktionen T_1, T_2, T_3 und T_4 verletzen das *strikte 2-Phasen* Sperrprotokoll? Begründen Sie kurz Ihre Antwort.

T_1	T_2	T_3	T_4
b_1	b_2	b_3	b_4
		$\mathrm{SL}_3(A)$	
			$\mathrm{XL}_4(B)$
	$\mathrm{SL}_2(A)$		
		$r_3(A)$	
			$w_4(B)$
$r_1(A)$			
$\mathrm{XL}_1(C)$			
$\mathrm{SL}_1(A)$			
			$\mathrm{SL}_4(D)$
	$\mathrm{relSL}_2(A)$		
			$r_4(D)$
			$\mathrm{relSL}_4(D)$
		$\mathrm{XL}_3(D)$	
		$w_3(D)$	
			$w_4(B)$
			$\mathrm{relXL}_4(B)$
	$\mathrm{XL}_2(B)$		
	$w_2(B)$		
	$\mathrm{relXL}_2(B)$		
	c_2		
		$\mathrm{relSL}_3(A)$	
		$\mathrm{relXL}_3(D)$	
		c_3	
			c_4
$\mathrm{relSL}_1(A)$			
$\mathrm{relXL}_1(C)$			
c_1			

Die folgenden Transaktionen verletzen das strikte 2-Phasen Sperrprotokoll:

T_1, T_2, T_4 .....

Begründung:

T_1 greift auf A zu ohne eine .....
entsprechende Sperre zu besitzen. .....
T_2 und T_4 geben beide ihre .....
Sperren nicht erst beim Commit .....
zurück, sondern führen noch .....
Aktionen aus, nachdem sie bereits .....
Sperren zurückgegeben haben. .....
T_2 verletzt darüber hinaus sogar .....
das Zwei-Phasen Sperrprotokoll, da .....
Sperren angefordert werden nachdem .....
bereits eine Sperre freigegeben wurde. ..

.....

.....

.....

Für die Aufgaben 4 – 6 gilt die Datenbankbeschreibung auf diesem Blatt.

Aufgabe 4: Erstellen eines Datenbankschemas mittels SQL (12)

Sie sind gefragt eine Datenbank zu entwerfen, um sich einen Überblick über die Nachfolger- und Vorläufer-Beziehungen moderner Film-Reihen zu verschaffen.

Folgendes Schema ist gegeben:
film(titel, jahr, genre, star: *person.name*)
person(name, lieblingsfilm: *film.titel*)
reihe(vorlaeufer: *film.titel*, nachfolger: *film.titel*)

Filme werden über ihren Titel identifiziert und haben eine `Person` die als Star des Films gilt. Es werden zusätzlich Informationen zum Erscheinungsjahr und Genre gespeichert. Dabei muss sichergestellt werden, dass nur Filme mit einem Erscheinungsjahr nach 1890 gespeichert werden können. Personen werden über ihren Namen identifiziert. Außerdem hat jede Person einen Lieblingsfilm.

Weiters gibt es Filmreihen (in der Tabelle `reihe` abgebildet). Dabei haben Filme beliebig viele Vorlaeufer und Nachfolger, also Filme die in der Reihe davor oder danach spielen. Sie können annehmen, dass keine Zyklen in dieser Relation vorkommen.

Geben Sie die nötigen SQL Statements an, um obiges Schema (inklusive aller Konsistenzbedingungen) anzulegen. Sie können dabei entsprechende (einfache) Datentypen für die Attribute wählen.

```
CREATE TABLE film(
          titel   VARCHAR(100),
          jahr    INTEGER       NOT NULL CHECK(jahr > 1890),
          genre   VARCHAR(100)  NOT NULL,
          star    VARCHAR(100)  NOT NULL,
          PRIMARY KEY(titel));

CREATE TABLE person(
          name           VARCHAR(100),
          lieblingsfilm  VARCHAR(100)  NOT NULL,
          PRIMARY KEY(name),
          FOREIGN KEY(lieblingsfilm) REFERENCES film(titel));

CREATE TABLE reihe(
          vorlaeufer    VARCHAR(100),
          nachfolger    VARCHAR(100),
          PRIMARY KEY (vorlaeufer,nachfolger),
          FOREIGN KEY (vorlaeufer)  REFERENCES film(titel),
          FOREIGN Key (nachfolger) REFERENCES film(titel));

ALTER TABLE film  ADD CONSTRAINT film_star
      FOREIGN KEY (star)  REFERENCES  person(name)
      DEFERRABLE INITIALLY DEFERRED;
```

Hinweis: Achten Sie bei den Statements auf die Reihenfolge.

Aufgabe 5: Rekursive Abfragen (12)

Gegeben ist die folgende Rekursive Abfrage auf dem Datenbank-Schema des vorherigen Beispiels:

```
WITH RECURSIVE T(titel) AS (
  SELECT  Nachfolger
  FROM    Reihe
  WHERE   Vorläufer ='Iron Man 3'
UNION ALL
  SELECT  Nachfolger
  FROM    T JOIN  (Reihe JOIN Film ON Nachfolger = Film.titel)
          ON Vorläufer = T.titel
  WHERE   Genre = 'Science-Fiction' )

SELECT  titel,jahr
FROM    Film NATURAL JOIN T;
```

Werten Sie diese Abfrage auf der Datenbank-Instanz, die auf der letzten Seite angegeben ist, aus:

name	jahr
Avengers - Age of Ultron	2015
Captain America - Civil War	2016
Spider-Man - Homecoming	2017
Black Panther	2018
Ant-Man and the Wasp	2018

Aufgabe 6: PL/SQL Trigger (11)

Erstellen Sie einen PL/pgSQL Trigger `trFavFilm`, der vor dem Löschen eines Eintrags aus der `Film`-Tabelle für jede Zeile die gelöscht werden soll, die Funktion `newFavFilm` aufruft.

Diese Funktion soll sicherstellen, dass jede Person einen neuen Lieblingsfilm bekommt, falls ihr Lieblingsfilm gelöscht wird. Daher, wenn Film F gelöscht wird, soll folgendes passieren:

- Für **jede** Person deren Lieblingsfilm F war soll ein neuer Lieblingsfilm gesetzt werden.
- Der neue Lieblingsfilm muss aus den Filmen (ohne F) gewählt werden, in denen die Person der Star war.
- Wenn kein solcher Film existiert, soll das Löschen abgebrochen werden und keine Änderungen an der Datenbank vorgenommen werden.

```
CREATE FUNCTION newFavFilm() RETURNS TRIGGER AS $$
DECLARE
    rec RECORD;
    newfilm VARCHAR(100);
BEGIN
    FOR rec in (SELECT name FROM Person WHERE lieblingsfilm = OLD.titel)
    LOOP
        SELECT name INTO newfilm FROM Film WHERE star=rec.name and titel != OLD.titel;
        IF newfilm IS NULL THEN
           RETURN NULL;
        END IF;
        UPDATE Personal SET lieblingsfilm=newfilm WHERE name=rec.name;
    END LOOP;

    RETURN OLD;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trFavFilm BEFORE DELETE ON Film
       FOR EACH ROW EXECUTE PROCEDURE newFavFilm();
```

Gesamtpunkte: 70

Sie können diesen Zettel abtrennen und brauchen ihn nicht abgeben!

Diesen Zettel daher bitte nicht beschriften! (Lösungen auf diesem Zettel werden nicht gewertet!)

Beispielinstanz für Aufgabe 5:

Film

titel	jahr	genre	star
The Incredible Hulk	2008	Action	Edward Norton
Iron Man 2	2010	Science-Fiction	Robert Downey Jr.
Captain America	2011	Action	Chris Evans
The Avengers	2012	Action	Robert Downey Jr.
Iron Man 3	2013	Science-Fiction	Robert Downey Jr.
Thor - The Dark Kingdom	2013	Action	Chris Hemsworth
Guardians of the Galaxy	2014	Action	Chris Pratt
Avengers - Age of Ultron	2015	Science-Fiction	Robert Downey Jr.
Ant-Man	2015	Science-Fiction	Paul Rudd
Captain America - Civil War	2016	Science-Fiction	Chris Evans
Guardians of the Galaxy Vol.2	2017	Science-Fiction	Chris Pratt
Spider-Man - Homecoming	2017	Science-Fiction	Tom Holland
Thor - Ragnarok	2017	Action	Chris Hemsworth
Black Panther	2018	Science-Fiction	Chadwick Boseman
Avengers - Infinity War	2018	Action	Robert Downey Jr.
Ant-Man and the Wasp	2018	Science-Fiction	Paul Rudd

Reihe

vorlaeufer	nachfolger
Iron Man 2	The Avengers
The Incredible Hulk	The Avengers
Captain America	The Avengers
The Avengers	Iron Man 3
The Avengers	Thor - The Dark Kingdom
Guardians of the Galaxy	Guardians of the Galaxy Vol.2
Iron Man 3	Avengers - Age of Ultron
Thor - The Dark Kingdom	Avengers - Age of Ultron
Captain America - Civil War	Spider-Man - Homecoming
Captain America - Civil War	Ant-Man and the Wasp
Captain America - Civil War	Black Panther
Avengers - Age of Ultron	Captain America - Civil War
Avengers - Age of Ultron	Guardians of the Galaxy
Avengers - Age of Ultron	Thor - Ragnarok
Ant-Man	Captain America - Civil War
Spider-Man - Homecoming	Avengers - Infinity War
Black Panther	Avengers - Infinity War
Thor - Ragnarok	Avengers - Infinity War
Guardians of the Galaxy Vol.2	Avengers - Infinity War

Person

name	lieblingsfilm
Robert Downey Jr.	Iron Man 2
Edward Norton	Ant-Man
Chris Hemsworth	Thor - The Dark Kingdom
Chris Evans	Captain America - Civil War
Chris Pratt	Guardians of the Galaxy
Paul Rudd	The Avengers
Tom Holland	Iron Man 3
Benedict Cumberbatch	Avengers - Infinity War
Chadwick Boseman	The Incredible Hulk