# 1 From SQL to the Logical Plan (5 Points)

In this exercise, we want to optimise a SQL query based on the following schemas using the rules presented in the lecture.

[persons] : {[pid: int, name: varchar, birth_year: int]}
[libraries] : {[lid: int, city: varchar]}
[members] : {[mid: (persons), favorite_author: (authors), late_fees: int]}
[membership] : {[member: (members), library: (libraries)]}
[authors] : {[aid: (persons), salary: int]}
[books] : {[bid: int, author: (authors), title: varchar, year: int, genre: varchar]}
[borrow] : {[member: (members), library: (libraries), book: (books), borrow_date: date, due_date: date]}
[reserve] : {[member: (members), library: (libraries), book: (books), reservation_date: date]}

```
SELECT favorite_author, reservation_date, year
FROM   books, reserve, members
WHERE  book = bid
       AND member = mid
       AND late_fees > 20
       AND title = 'The Da Vinci Code'
       AND reservation_date = '24.02.2024';
```

(a) Translate the SQL query canonically into a relational algebra expression. Use only projections, selections and Cartesian products. (1 Point)

(b) Draw the logical plan of the query as a tree. (1 Point)

(c) Apply the rules for heuristic query optimisation known from the lecture and the notebook Rule-based Optimization.ipynb and draw the optimised logical plan of the query as a tree. (3 Points)

## 1 From SQL to the Logical Plan (5 Points)

In this exercise, we want to optimise a SQL query based on the following schemas using the rules presented in the lecture.

[persons] : {[pid: int, name: varchar, birth_year: int]}
[libraries] : {[lid: int, city: varchar]}
[members] : {[mid: (persons), favorite_author: (authors), late_fees: int]}
[membership] : {[member: (members), library: (libraries)]}
[authors] : {[aid: (persons), salary: int]}
[books] : {[bid: int, author: (authors), title: varchar, year: int, genre: varchar]}
[borrow] : {[member: (members), library: (libraries), book: (books), borrow_date: date, due_date: date]}
[reserve] : {[member: (members), library: (libraries), book: (books), reservation_date: date]}

```
SELECT favorite_author, reservation_date, year
FROM   books, reserve, members
WHERE  book = bid
       AND member = mid
       AND late_fees > 20
       AND title = 'The Da Vinci Code'
       AND reservation_date = '24.02.2024';
```

(a) Translate the SQL query canonically into a relational algebra expression. Use only projections, selections and Cartesian products. (1 Point)

(b) Draw the logical plan of the query as a tree. (1 Point)

(c) Apply the rules for heuristic query optimisation known from the lecture and the notebook Rule-based Optimization.ipynb and draw the optimised logical plan of the query as a tree. (3 Points)


```
SELECT favorite_author, reservation_date, year
FROM   books, reserve, members
WHERE  book = bid
       AND member = mid
       AND late_fees > 20
       AND title = 'The Da Vinci Code'
       AND reservation_date = '24.02.2024';
```

(a) Translate the SQL query canonically into a relational algebra expression. Use only projections, selections and Cartesian products. (1 Point)

(b) Draw the logical plan of the query as a tree. (1 Point)

(c) Apply the rules for heuristic query optimisation known from the lecture and the notebook Rule-based Optimization.ipynb and draw the optimised logical plan of the query as a tree. (3 Points)

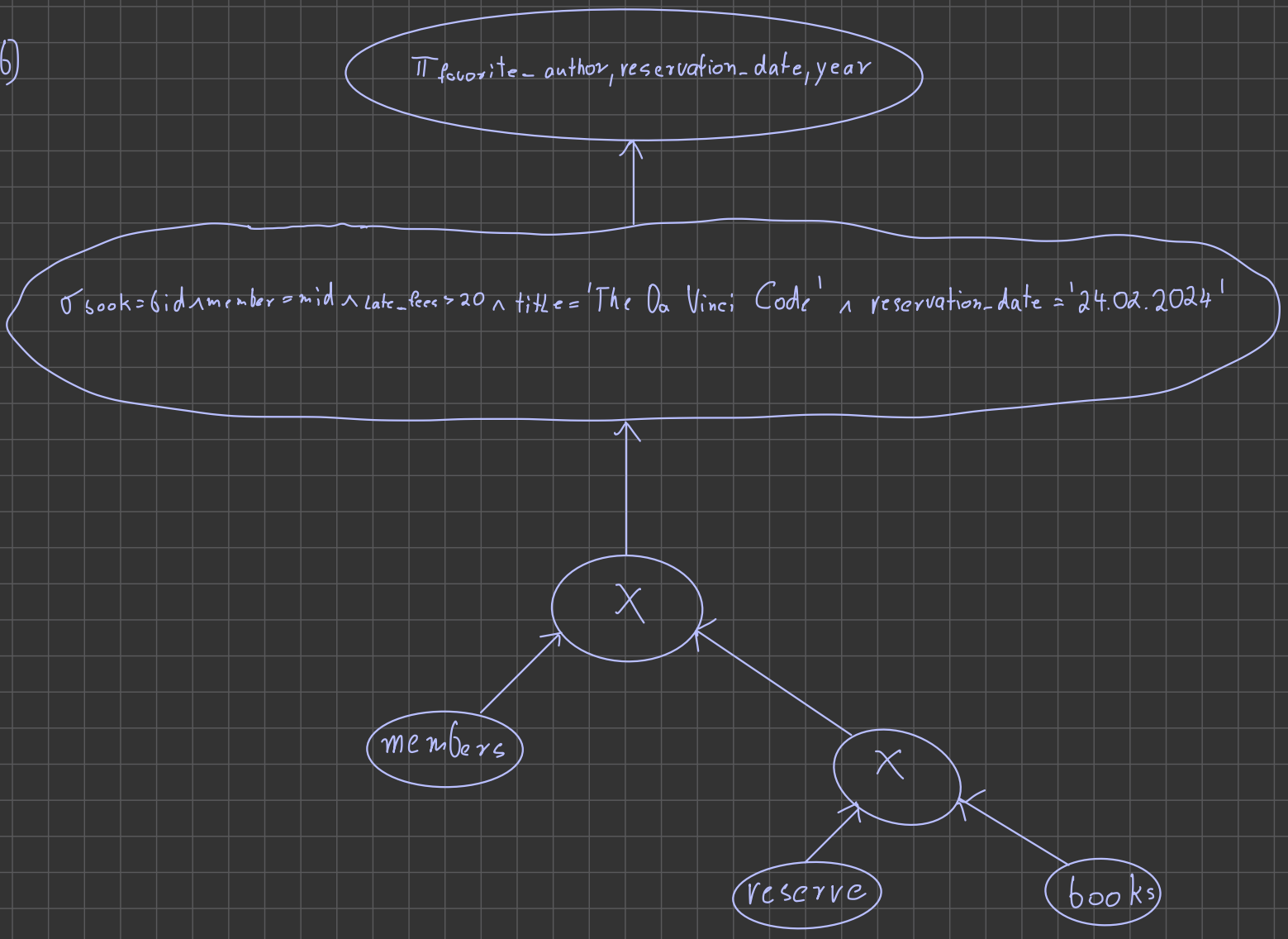

a)

$R_1 := (members \times (reserve \times books))$

$R_2 := \sigma_{book=bid \wedge member=mid \wedge late\_fees>20 \wedge title='The\ Da\ Vinci\ Code' \wedge reservation\_date='24.02.2024'} R_1$

$\pi_{favorite\_author,\ reservation\_date,\ year} R_2$

b)



## 1 From SQL to the Logical Plan (5 Points)

In this exercise, we want to optimise a SQL query based on the following schemas using the rules presented in the lecture.

[persons] : {[pid: int, name: varchar, birth_year: int]}
[libraries] : {[lid: int, city: varchar]}
[members] : {[mid: (persons), favorite_author: (authors), late_fees: int]}
[membership] : {[member: (members), library: (libraries)]}
[authors] : {[aid: (persons), salary: int]}
[books] : {[bid: int, author: (authors), title: varchar, year: int, genre: varchar]}
[borrow] : {[member: (members), library: (libraries), book: (books), borrow_date: date, due_date: date]}
[reserve] : {[member: (members), library: (libraries), book: (books), reservation_date: date]}

```
SELECT favorite_author, reservation_date, year
FROM   books, reserve, members
WHERE  book = bid
       AND member = mid
       AND late_fees > 20
       AND title = 'The Da Vinci Code'
       AND reservation_date = '24.02.2024';
```

(a) Translate the SQL query canonically into a relational algebra expression. Use only projections, selections and Cartesian products. (1 Point)

(b) Draw the logical plan of the query as a tree. (1 Point)

(c) Apply the rules for heuristic query optimisation known from the lecture and the notebook Rule-based Optimization.ipynb and draw the optimised logical plan of the query as a tree. (3 Points)

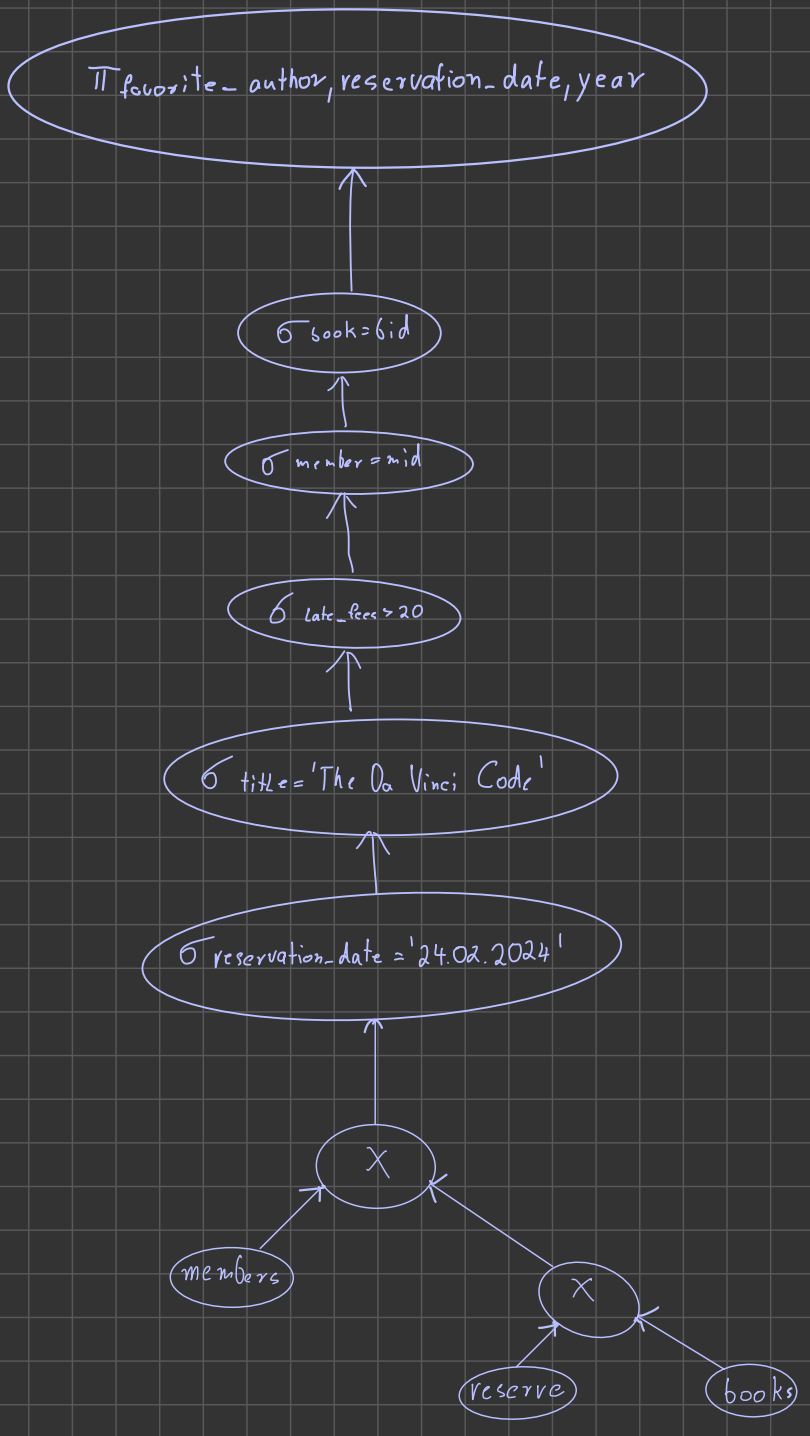

```
SELECT favorite_author, reservation_date, year
FROM   books, reserve, members
WHERE  book = bid
       AND member = mid
       AND late_fees > 20
       AND title = 'The Da Vinci Code'
       AND reservation_date = '24.02.2024';
```

(a) Translate the SQL query canonically into a relational algebra expression. Use only projections, selections and Cartesian products. (1 Point)

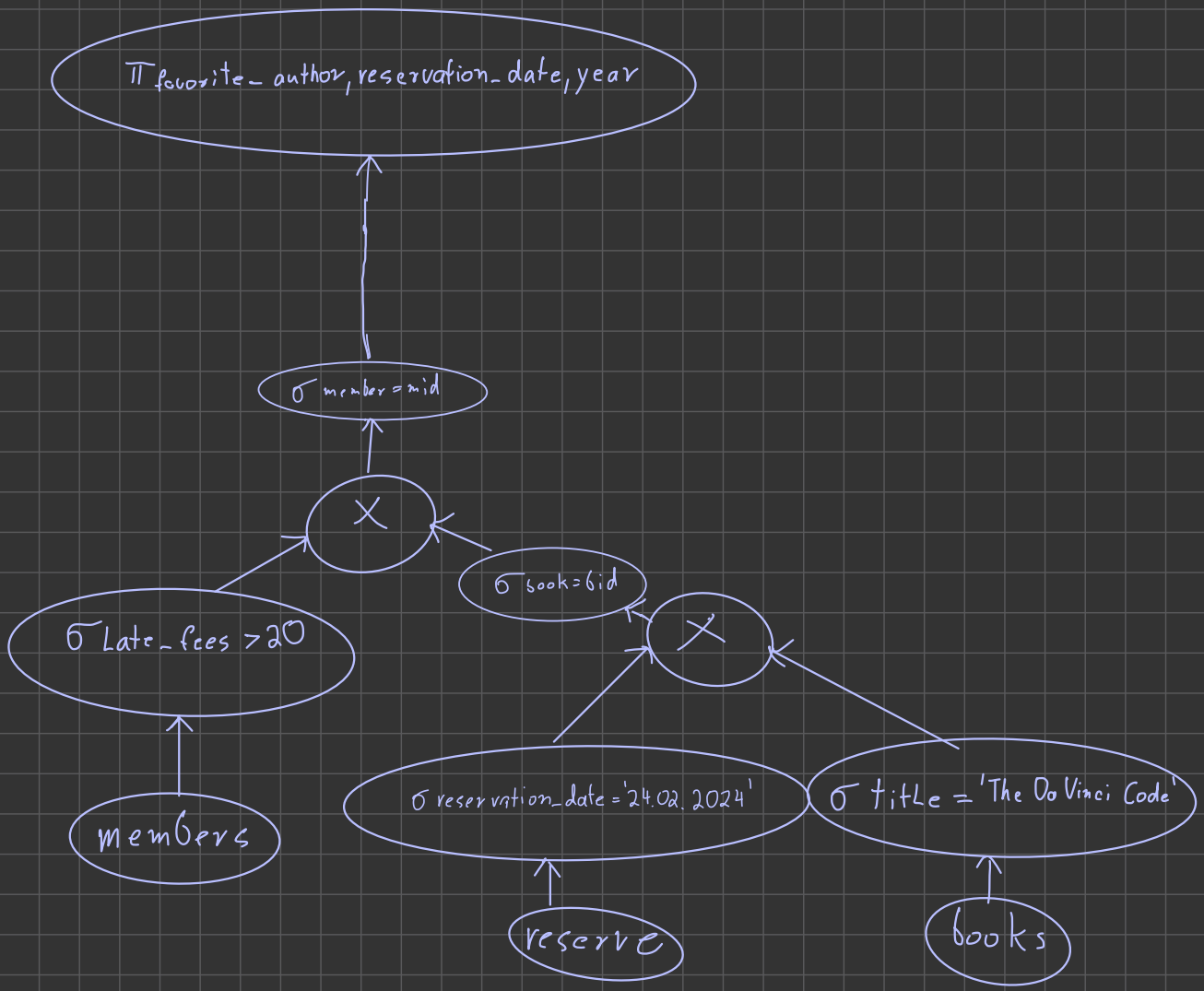
(b) Draw the logical plan of the query as a tree. (1 Point)

(c) Apply the rules for heuristic query optimisation known from the lecture and the notebook Rule-based Optimization.ipynb and draw the optimised logical plan of the query as a tree. (3 Points)

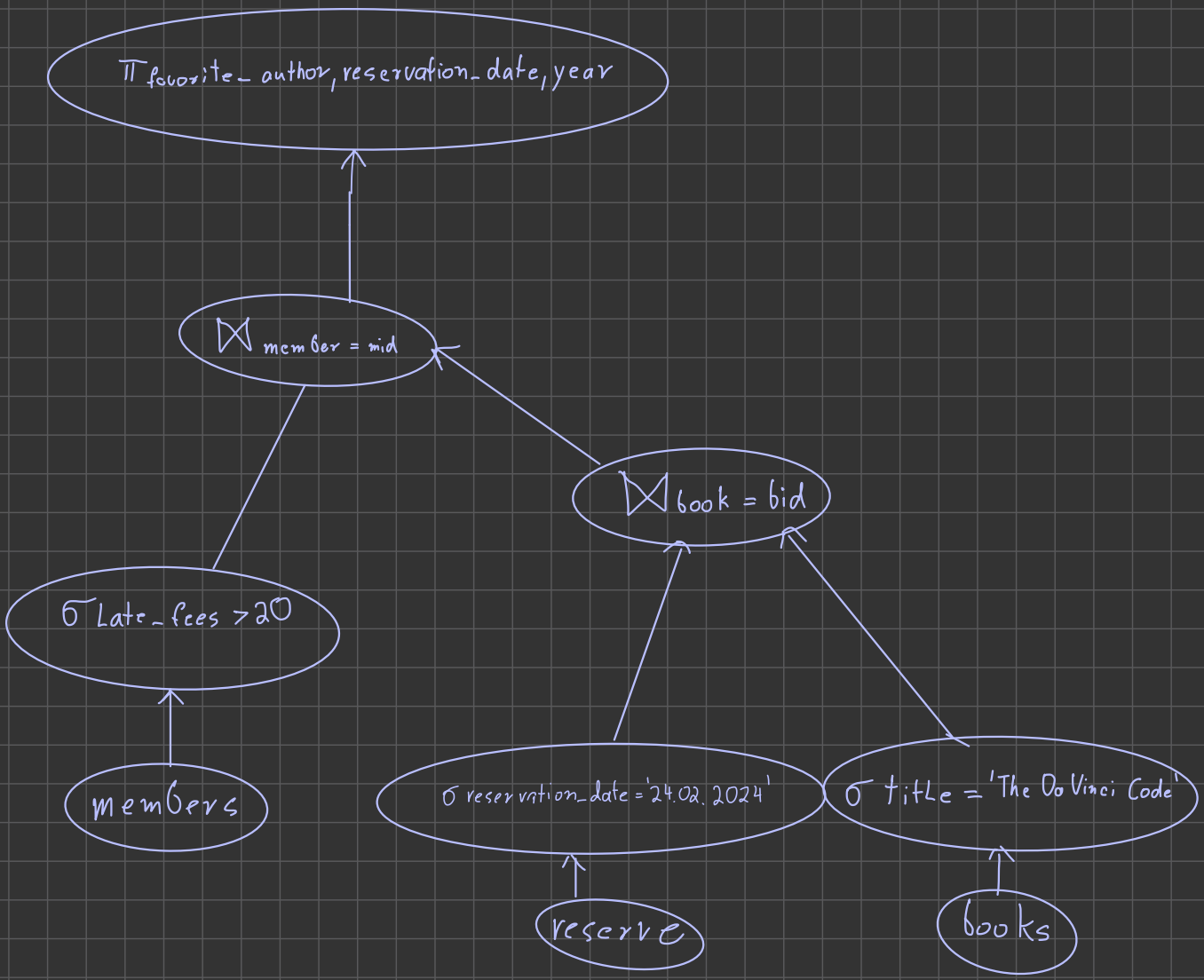

c) 1. Break up and selections



c) 2. Push down selections



c) 3. Replace a join style selection on top of a cartesian product by a theta join



c) 4. Insert projections