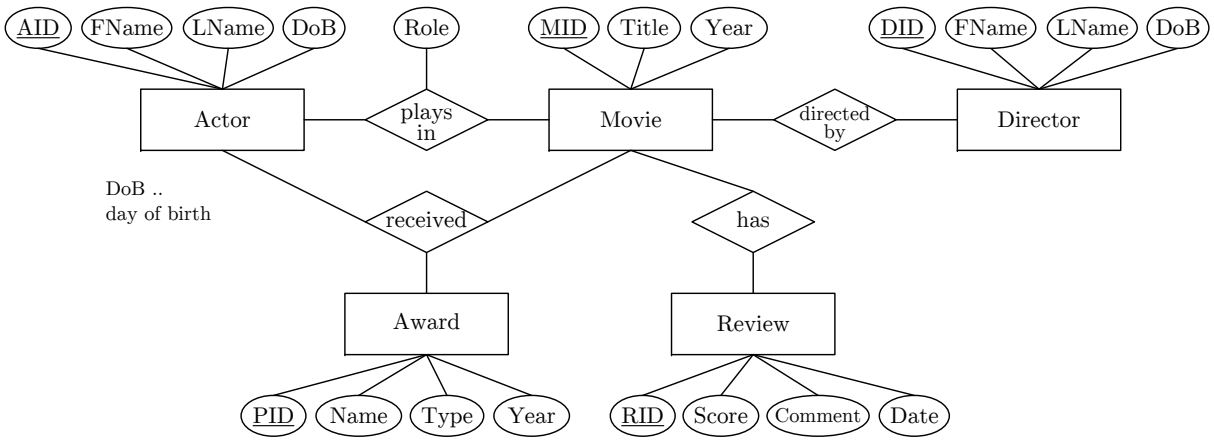
Univ.-Prof. Dr.-Ing. Matthias Boehm
Graz University of Technology
Computer Science and Biomedical Engineering
Institute of Interactive Systems and Data Science
BMK endowed chair for Data Management

July 07, 2022

Exam INF.01017UF Data Management (Summer 2022, V2a)
Exam 706.010 Databases (Summer 2022, V2b)

Important notes: The working time is 90min, and lecture materials or any kind of mobile devices are not allowed. Please, make sure to put your *name* and *matriculation number* on the top right of the first page of the task description, and each additional piece of paper. You may give the answers in English or German, written directly into the task description.

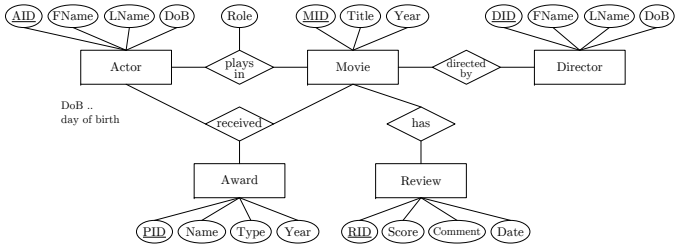
Task 1 Data Modeling (25 points)



(a) Given the Entity-Relationship (ER) diagram above, specify the cardinalities in Modified Chen notation based on the following information. (**9 points**)

- An actor may play roles in an arbitrary number of movies (including none), and every movie has a cast of at least one but potentially many actors.
- A movie is directed by exactly one director, and a single director might produce (i.e., direct) an arbitrary number of movies.
- A movie review (with score, text comment, and date) refers to exactly one movie, but there can be 0, 1, or many reviews per movie.
- A single actor may receive multiple—specifically up to 8—awards (e.g., best actress, best supporting actress) for a specific, single movie. An actor receives such a single award for exactly one movie though. A single award (e.g., best ensemble) for a single movie can be awarded to one or multiple actors.

(b) Map the given ER diagram into a relational schema in third normal form, including data types, primary keys, and foreign keys. Your schema should also ensure that each movie has an associated director, and each review refers to a movie. Note that you only need to provide the final schema and there is no need to explain the normal forms. (**12 points**)



(c) Assume an independent relation Movies(MID, Title, Year) with MID and Title each being unique and defined (not null). List below all super keys, all candidate keys, and select an appropriate primary key. Use (a,b,c) to indicate compound keys. (**4 points**)

- Super Keys:
- Candidate Keys:
- Primary Key:

Task 2 Structured Query Language (30 points)

Movies

			[min]	[Mio $]	[Mio $]	
MID	**Title**	**Year**	**Length**	**Budget**	**Revenue**	**GID**
1	The Matrix	1999	136	63	455	2
3	Hangover	2009	100	35	470	1
2	Fast and Furious	2001	106	40	210	3
7	Passengers	2016	116	130	300	2
4	Horrible Bosses	2011	98	35	210	1
5	The Hunger Games	2012	142	80	700	2
6	Draft Day	2014	110	25	30	4
8	The Post	2017	116	50	180	5

Genres

GID	**Name**
1	Comedy
6	Romance
2	Science Fiction
3	Action
4	Sports Drama
5	Historical Drama
7	Documentary

(a) Given the Movies and Genres tables above, compute the results for the following three queries: (**15 points**)

```
Q1: SELECT M.Title, M.Year, G.Name
      FROM Movies M, Genres G
      WHERE M.GID = G.GID
        AND (G.Name LIKE '% Drama'
        OR M.Length BETWEEN 130 AND 140)


Q2: SELECT Title, Year
      FROM Movies WHERE Year > 2010
    INTERSECT
    SELECT Title, Year
      FROM Movies WHERE Revenue > 250



Q3: SELECT Name, round(avg(Revenue)) --avg=sum/count
      FROM Movies M JOIN Genres G ON (M.GID=G.GID)
      GROUP BY Name
      ORDER BY avg(Revenue) DESC
```

(b) Given the Movies and Genres tables above, write SQL queries to answer the following questions (in a way that is independent of the shown data): (**15 points**)

- Q4: Which movies belong to the genre "Science Fiction" (return the Title and Year, sorted in ascending order of Title)?

- Q5: Which movie from the years 2005-2015 (both inclusive) yielded the maximum Revenue (return the Title of this movie and its Revenue)?

- Q6: Compute the number of movies associated with each genre, including genres without any movies (return the genre Name, and count)?

Task 3 Query Processing (17 points)

(a) Given a relation $R(x, y, z)$ with four tuples (a, b, c), (d, e, f), (a, b, d), and (d, e, g), indicate in the table below for each relational algebra expression (row) the number of output tuples in set and bag (multiset) semantics, respectively. (**6 points**)

RA Expression	**Set Semantics**	**Bag Semantics**
$\pi_{x,y}(R)$		
$\sigma_{x=a \ \vee \ z=f}(R)$		
$R \cup R$		

(b) Draw a logical query tree for the following query. (**5 points**)

```
Q7: SELECT G.Name, count(*)
      FROM Movies M, Genres G
      WHERE M.GID = G.GID
        AND M.Revenue > 200
      GROUP BY G.Name
      HAVING count(*) >= 2
      ORDER BY count(*) DESC
```

(c) Describe the conceptual ideas of a nested-loop join, and a hash join. Furthermore, assume $R \bowtie S$ with cardinalities $N = |R|$ and $M = |S|$, and enter the space and time complexity of these operators (in the open-next-close iterator model) in the table below. (**6 points**)

Operator	**Time Complexity**	**Space Complexity**
Nested Loop Join		
Hash Join		

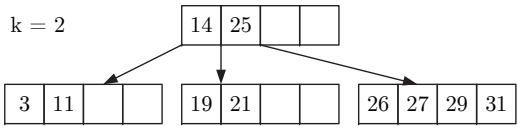
Task 4 Transaction Processing (8 points)

(a) Explain the concept of a database transaction log, and how it helps to ensure Atomicity and Durability of changes made by uncommitted and committed transactions on failures. (**6 points**)

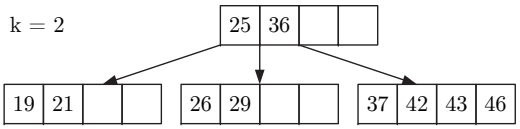
(b) Briefly explain the concept of a transaction isolation level. (**2 points**)

Task 5 Physical Design (20 points)

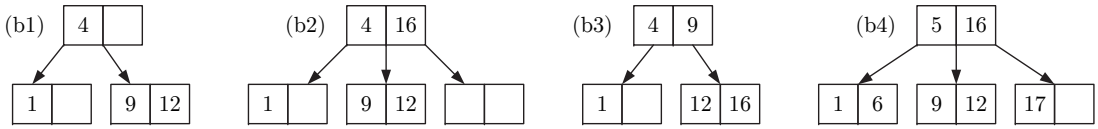
(a) Given the B-tree with k=2 below, insert the keys 7 and 37, and draw the resulting final B-tree. (**5 points**)



(b) Given the B-tree with k=2 below, delete the keys 21 and 43, and draw the resulting final B-tree. (**5 points**)



(c) Which of the following trees are valid—i.e., satisfy the constraints of—B-trees with k=1. Mark each tree as valid (✓) or invalid (×) and name the violations. (**4 points**)



(d) Recall the table `Movies` from Task 2 and perform a horizontal and vertical partitioning, respectively. Specifically, provide—for both scenarios—relational algebra expressions for partitioning `Movies` into two fragments `Movies1` and `Movies2`, as well as its subsequent reconstruction from the two fragments. (**6 points**)

- Horizontal Partitioning (row partitions):
  - Movies1 :=
  - Movies2 :=
  - Movies :=

- Vertical Partitioning (column partitions):
  - Movies1 :=
  - Movies2 :=
  - Movies :=