

Collection of database exam solutions

Rasmus Pagh

October 19, 2011

This is a supplement to the collection of database exams used in the course *Introduction to Database Design*, which includes answers. The idea is that it can be used to:

- Check your own solutions against.
- Get an impression of what is required for a written solution to be considered complete.

Data Storage and Formats

IT University of Copenhagen

January 5, 2009

This exam is a translation, by Michael Magling, of an original Danish language exam. It consists of 6 problems with a total of 15 questions. The weight of each problem is stated. You have 4 hours to answer all questions. The complete assignment consists of 11 pages (including this page). **It is recommended to read the problems in order**, but it is not important to solve them in order.

If you cannot give a complete answer to a question, try to give a partial answer.

The pages in the answer must be ordered and numbered, and be supplied with name, CPR-number and course code (BDLF). Write only on the front of sheets, and order them so that the problems appear in the correct order.

“KBL” refers to the set in the course book “Database Systems - an application approach, 2nd edition”, by Michael Kifer, Arthur Bernstein and Philip M. Lewis.

All written aids are allowed.

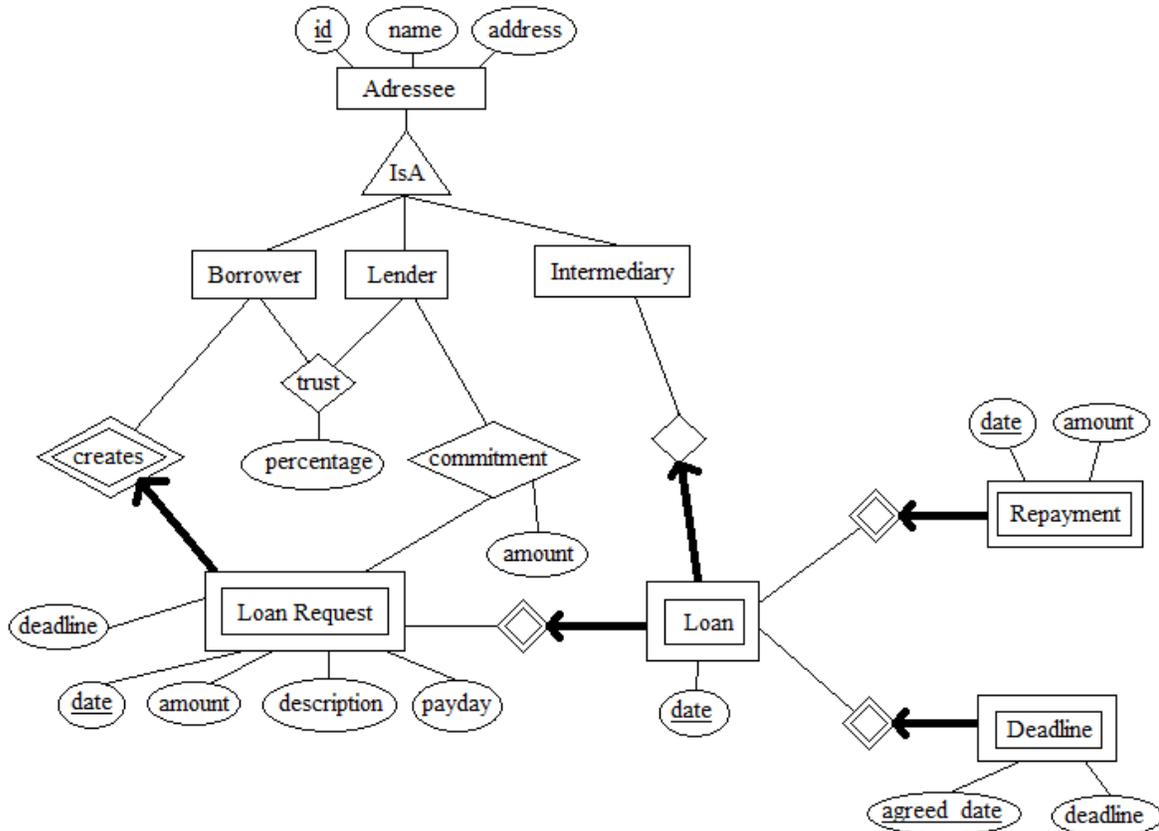
1 Data modeling (25%)

Micro loans are small loans, which is beginning to gain popularity especially among borrowers in developing countries. The idea is to bring venture lenders together using information technology. Typically, the loans will be used to finance startup or development of the borrower's company, so that there is a realistic chance for repayment. The money in a loan can, unlike traditional loans, come from many lenders. In this problem, you must create an E-R model that describes the information necessary to manage micro loans. The following information form the basis for creating the model:

- Each borrower and lender must be registered with information about name and address.
- A loan starts with a loan request, which contains information about when the loan should at latest be granted, The total amount being discussed (US-dollars), and how long the payback period is. Also, a description is included of how the money will be used. The rent on the payment is calculated in the loan amount, which is to say, the full amount is not paid .
- Lenders can commit to an optional portion of the total amount of a loan request.
- When the commitments for the loan request covers the requested amount, the request is converted to a loan. If not enough commitments can be reached, the loan request is cancelled. A borrower can have more than one request, and more than one loan at a time, but can at most make one request per day.
- The loan is paid through an “intermediary”, typically a local department of a charity, who has a name and an address.
- The borrower chooses when he or she will make a payment. Every payment must be registered in the database with an amount and a date (at most one payment per loan per day). The lenders share the repayment based on how large a part of the loan they are responsible for.
- If the loan is not repaid before the agreed upon deadline, a new date is agreed. The database must not delete the old deadline, but save the history (the deadline can be overridden multiple times).
- Each lender can for each burrower save a “trust”, which is a number between 0 and 100 that determines the lender's evaluation of the risk of lending money to that person. The number must only be saved for the borrowers, for whom there has been made such an evaluation.

a) Make an E-R model for the data described above. If you make any assumptions about data that doesn't show from the problem, they must be described. Use the E-R notation from KBL. Put an emphasis on having the model express as many properties about the data as possible, for instance participation constraints.

Example answer:



b) Make a relational data model for micro loans:

- Describe at least **two** of the relations using SQL DDL (make reasonable assumptions about data types), and
- state the relation schemas for the other relations.

The emphasis is if there is a correlation between the relational model and the E-R diagram from a), along with primary key and foreign key constraints being stated for all relation. It is *not* necessary to state CHECK constraints and the like.

Example answer: It is assumed, that borrowers, lenders, and intermediaries are disjoint entities. Below MySQLs ENUM type is used, which is not part of the syllabus.

```
CREATE TABLE Adressee (
```

```

    id INT PRIMARY KEY,
    type ENUM('borrower', 'lender', 'intermediary'),
    name VARCHAR(50),
    address VARCHAR(50)
);

CREATE TABLE Trust (
    borrower INT REFERENCES Adressee(id),
    lender INT REFERENCES Adressee(id),
    percentage INT,
    PRIMARY KEY (borrower,lender)
);

CREATE TABLE LoanRequest (
    id INT REFERENCES Adressee(id),
    date DATE,
    amount INT,
    description VARCHAR(1000),
    payday DATE,
    deadline DATE,
    PRIMARY KEY (id,date)
);

CREATE TABLE Commitment (
    lender INT REFERENCES Adressee(id),
    borrower INT,
    loanrequestDate DATE,
    FOREIGN KEY (borrower,loanrequestDate) REFERENCES LoanRequest(id,dato),
    amount INT,
    PRIMARY KEY (lender, borrower, loanrequestDate,amount)
);

CREATE TABLE Loan (
    id INT,
    RequestDate DATE,
    date DATE,
    intermediary REFERENCES Adressee(id),
    FOREIGN KEY (id,RequestDate) REFERENCES LoanRequest(id,date),
    PRIMARY KEY(date,id,RequestDate)
);

CREATE TABLE Repayment (
    id INT,
    date DATE,
    RequestDate DATE,

```

```

    amount INT,
    FOREIGN KEY (id,RequestDate) REFERENCES LoanRequest(id,date),
    PRIMARY KEY (date,id,RequestDate)
);

CREATE TABLE Deadline (
    id INT,
    agreedDate DATE,
    RequestDate DATE,
    deadline DATE,
    FOREIGN KEY (id,RequestDate) REFERENCES LoanRequest(id,date),
    PRIMARY KEY (agreedDate,id,RequestDate)
);

```

2 XML (20%)

Consider the following XML document, `loaners.xml`:

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet href="mystylesheet.xsl" type="text/xsl"?>
<microloans>
<loaner>
  <name>
    <first>Nandela</first>
    <last>Melson</last>
  </name>
  <address>Freedom Way 1, 23456 Johannesburg, South Africa</address>
  <loan>
    <amount>1000</amount>
    <payout-date>1990-01-01</payout-date>
    <repayment amount="100" date="1991-01-01"/>
    <repayment amount="100" date="1992-01-01"/>
  </loan>
  <loan>
    <amount>500</amount>
    <payout-date>1993-01-01</payout-date>
    <repayment amount="100" date="1991-01-01"/>
  </loan>
</loaner>
<loaner>
  <name>
    <first>Majeev</first>
    <last>Rotwani</last>
  </name>

```

```
<address>Circle Strait 8, 98764 Bumbai, India</address>
</loaner>
</microloans>
```

a) Write an XPath expression that returns all of the name (**name** elements) in `loaners.xml`. Emphasis is on if the expression also works on other, similar, XML documents.

Example answer:

```
//name
```

b) Write an XPath expression that returns all the names of borrowers, who have (had) at least one loan, which is to say, where there is a **loan** element. Emphasis is on if the expression also works on other, similar, XML documents.

Example answer:

```
//loaner[loan]/name
```

Consider the following XSL stylesheet, `mystylesheet.xsl`:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="microloans">
  <html>
  <body>
    <xsl:apply-templates select="//loan"/>
  </body>
  </html>
</xsl:template>

<xsl:template match="loan">
  <xsl:apply-templates select="../name/last"/>,
  <xsl:apply-templates select="../name/first"/>:
  <xsl:apply-templates select="amount"/><br/>
</xsl:template>
</xsl:stylesheet>
```

c) State the result of running `mystylesheet.xsl` on `loaners.xml`.

Example answer:

```
<html>
<body>
  Melson, Nandela: 1000<br/>
```

```
Melson, Mandela: 500<br/>
</body>
</html>
```

d) Write an XQuery expression that for each borrower in `loaners.xml` computes the total amount, which is to say the sum of the numbers in the `amount` elements, minus the sum of the numbers in the `repayment` attribute of the `repayment` elements. The output must be valid XML that for each borrower states name and outstanding amount (in a `debt` element).

Example answer:

```
<loaners>
{
for $x in doc("loaners.xml")//loaner
return <loaner>
    {$x/name}
<debt> {fn:sum($x/loan/amount) - fn:sum($x/loan/repayment/@amount)} </debt>
</loaner>
}
</loaners>
```

3 Normalization (10%)

The following relation schema can be used to register information on the repayments on micro loans (see the text in the problem 1 for the explanation on micro loans, and the example on data about micro loans in problem 2).

`Repayment(borrower_id,name,address,loanamount,requestdate,repayment_date,request_amount)`

A borrower is identified with an unique `borrower_id`, and has only one address. Borrowers can have multiple simultaneous loans, but they always have different request dates. The borrower can make multiple repayments on the same day, but not more than one repayment per loan per day.

a) State a key (candidate key) for `Repayment`.

Example answer: `{borrower_id,requestdate,repayment_date}`

b) Make the normalization to BCNF. State for every step in the normalization, which functional dependency that causes it.

Example answer: Functional dependencies:

`borrower_id → name address`
`borrower_id requestdate → loanamount`

BCNF:

`Repayment1(borrower_id,name,address)`
`Repayment2(borrower_id,requestdate,loanamount)`
`Repayment3(borrower_id,requestdate,repayment_date,repayment_amount)`

4 SQL (25 %)

This problem is about writing SQL for the relation `Repayment` from problem 3:

`Repayment(borrower_id,name,address,loanamount,requestdate,repayment_date,repayment_amount)`

To solve the problem, the information from the description in problem 3 must be used.

a) Write an SQL request that returns all the tuples with information on repayments from the borrower with `id` equal to 42, and where the lent amount exceeds 1000 USD.

Example answer:

```
SELECT *
FROM Repayment
WHERE borrower_id=42 AND loanamount>1000;
```

b) Write an SQL request that for each address finds the total repaid amount for the address.

Example answer:

```
SELECT address, SUM(repayment_amount)
FROM Repayment
GROUP BY address;
```

c) Write an SQL request that finds all names which has a unique address, which to say is where there does not exist a tuple with a different name and same address.

Example answer:

```
SELECT name
FROM Repayment A
WHERE 1=
  (SELECT COUNT(DISTINCT name)
   FROM Repayment B
   WHERE A.address=B.address);
```

d) Write an SQL command, which deletes all information on ended loans, which is to say loans where the total repaid amount equals the lend amount.

Example answer:

```
DELETE FROM Repayment A
WHERE loanamount=
  (SELECT SUM(repayment_amount)
   FROM Repayment B
   WHERE B.borrower_id=A.borrower_id AND B.requestdate=A.requestdate);
```

5 Transactions (10 %)

Consider the following transactions, which uses explicit locking of tuples. Here ?1, ?2, ?3, ?4 is used to reference parameters that are substituted in.

1. SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SELECT * FROM R WHERE id=?1 FOR UPDATE;
SELECT * FROM S WHERE pk=?2 FOR UPDATE;
UPDATE R SET a=?3 WHERE id=?1;
UPDATE S SET b=?4 WHERE pk=?2;
COMMIT;
2. SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SELECT * FROM S WHERE pk=?1 FOR UPDATE;
SELECT * FROM R WHERE id=?2 FOR UPDATE;
UPDATE S SET d=?3 WHERE pk=?1;
UPDATE R SET c=?4 WHERE id=?2;
COMMIT;

a) Argue that there is a possibility for deadlocks, if the two transactions are run at the same time. State a specific sequence of locks, that leads to a deadlock.

Example answer: If the two transactions both locks the same tuples in R of S deadlocks are created for instance in the situation where transaction 1 locks R, and transaction 2 after that locks S.

b) Suggest a change of the transactions, so deadlocks can no longer be created, and give a short argument that this is in fact the case. Emphasis is on that the transactions keep their original effect.

Example answer: By changing the order of the rowlocks in one of the transaction, the problem is avoided (the two transactions locks in the same order, and as asuch, deadlocks cannot be created).

6 Indexing (10%)

We again look at the relation `Repayment` from problem 3 (un-normalized). Assume that the following four SQL commands are known to be frequent (with actual parameters substituted in for `?`):

1. `SELECT DISTINCT name, address
FROM Repayment
WHERE borrower_id = ?;`
2. `SELECT *
FROM Repayment
WHERE borrower_id = ? AND repayment_date > ?;`
3. `SELECT borrower_id, loanamount
FROM Repayment
WHERE loanamount BETWEEN ? AND ?;`
4. `INSERT INTO Request VALUES (?, ?, ?, ?, ?, ?, ?, ?);`

a) Suggest one or more indexes, taking into account of the above. State the indexed attributes for each index, along with the index type (primary or secondary). Argue shortly for your choices. Emphasis is on the suggested indexes supports the SQL commands as effectively as possible.

Example answer:

Primary index (B-tree) on `borrower_id`, `repayment_date` (used with 1 and 2).

Secondary index (B-tree) on `loanamount`, `borrower_id` (covering index from request 3).

Introduction to Databases

IT University of Copenhagen

January 16, 2006

This exam consists of 5 problems with a total of 16 questions. The weight of each problem is stated. You have 4 hours to answer all 16 questions. The complete assignment consists of 12 numbered pages (including this page), *plus an answer sheet to be used for several of the questions.*

If you cannot give a complete answer to a question, try to give a partial answer. You may choose to write your answer in Danish or English. Write only on the front of sheets, and remember to write your CPR-number on each page. Please start your answer to each question at the top of a *new* page. Please order and number the pages before handing in.

G UW refers to *Database Systems – The Complete Book* by Hector Garcia-Molina, Jeff Ullman, and Jennifer Widom, 2002.

All written aids are allowed / Alle skriftlige hjælpemidler er tilladt.

1 Database design (25%)

The academic world is an interesting example of international cooperation and exchange. This problem is concerned with modeling of a database that contains information on researchers, academic institutions, and collaborations among researchers. A researcher can either be employed as a professor or a lab assistant. There are three kinds of professors: Assistant, associate, and full professors. The following should be stored:

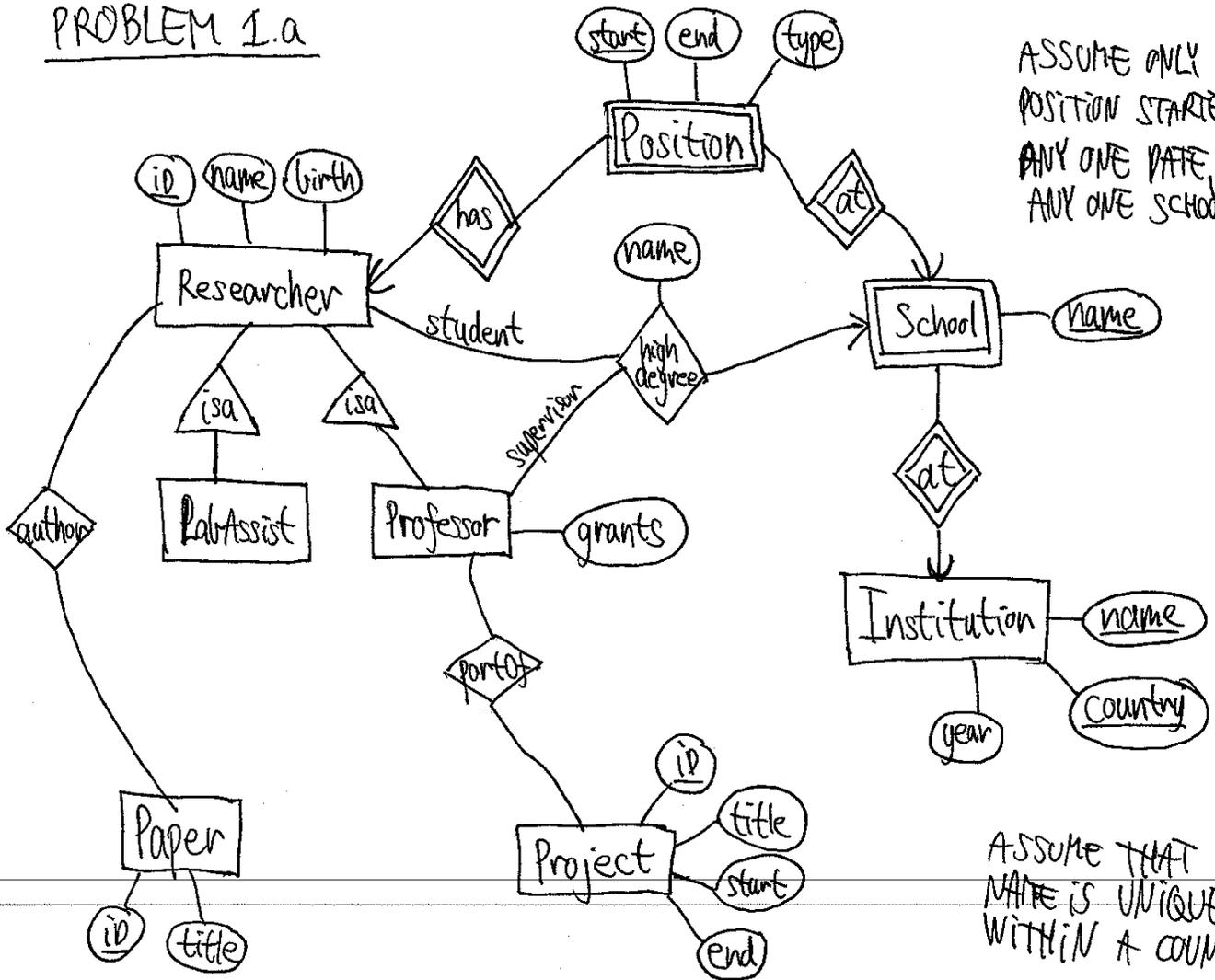
- For each researcher, his/her name, year of birth, and current position (if any).
- For each institution, its name, country, and inauguration year.
- For each institution, the names of its schools (*e.g. School of Law, School of Business, School of Computer Science,...*). A school belongs to exactly one institution.
- An employment history, including information on all employments (start and end date, position, and what school).
- Information about co-authorships, i.e., which researchers have co-authored a research paper. The titles of common research papers should also be stored.
- For each researcher, information on his/her highest degree (BSc, MSc or PhD), including who was the main supervisor, and at what school.
- For each professor, information on what research projects (title, start date, and end date) he/she is involved in, and the total amount of grant money for which he/she was the main applicant.

a) Draw an E/R diagram for the data set described above. Make sure to indicate all cardinality constraints specified above. The E/R diagram should not contain redundant entity sets, relationships, or attributes. Also, use relationships whenever appropriate. If you need to make any assumptions, include them in your answer.

Answer:

See next page.

PROBLEM 1.a



ASSUME ONLY ONE POSITION STARTED AT ANY ONE DATE AT ANY ONE SCHOOL

ASSUME THAT NAME IS UNIQUE WITHIN A COUNTRY

b) Convert your E/R diagram from question a) into relations, and write SQL statements to create the relations. You may make any reasonable choice of data types. Remember to include any constraints that follow from the description of the data set or your E/R diagram, including primary key and foreign key constraints.

Answer:

```
CREATE TABLE Researcher (  
  ID int PRIMARY KEY,  
  name VARCHAR(50),  
  birth INT  
);
```

```
CREATE TABLE Professor (  
  ID int REFERENCES Researcher(ID),  
  grants INT  
);
```

```
CREATE TABLE Paper (  
  ID int PRIMARY KEY,  
  title VARCHAR(50)  
);
```

```
CREATE TABLE Position (  
  researcherID int REFERENCES Researcher(ID),  
  start DATE,  
  end DATE,  
  schoolName VARCHAR(50),  
  institutionName VARCHAR(50),  
  FOREIGN KEY (schoolName,institutionName) REFERENCES School(name,insitutionName),  
  PRIMARY KEY (researcherID,start,schoolName,institutionName)  
);
```

```
CREATE TABLE Project (  
  ID int PRIMARY KEY,  
  title VARCHAR(50),  
  start DATE,  
  end DATE  
);
```

```
CREATE TABLE School (  
  name VARCHAR(50),  
  institutionName VARCHAR(50) REFERENCES Institution(name),  
  PRIMARY KEY (name, institutionName)  
);
```

```
CREATE TABLE Institution (  
name VARCHAR(50),  
country VARCHAR(50),  
year INT,  
PRIMARY KEY (name, country)  
);
```

```
CREATE TABLE Author (  
researcherID INT REFERENCES Researcher(ID),  
paperID INT REFERENCES Paper(ID),  
PRIMARY KEY (researcherID, paperID)  
);
```

```
CREATE TABLE PartOf (  
professorID INT REFERENCES Professor(ID),  
studentID INT REFERENCES Researcher(ID),  
PRIMARY KEY (professorID, projectID)  
);
```

```
CREATE TABLE HighDegree (  
professorID INT REFERENCES Professor(ID),  
projectID INT REFERENCES Project(ID),  
schoolName VARCHAR(50),  
institutionName VARCHAR(50),  
FOREIGN KEY (schoolName,institutionName) REFERENCES School(name,insitutionName),  
PRIMARY KEY (professorID, studentID, schoolName, institutionName)  
);
```

2 Normalization (15%)

We consider the following relation:

`Articles(ID, title, journal, issue, year, startpage, endpage, TR-ID)`

It contains information on articles published in scientific journals. Each article has a unique ID, a title, and information on where to find it (name of journal, what issue, and on which pages). Also, if results of an article previously appeared in a “technical report” (TR), the ID of this technical report can be specified. We have the following information on the attributes:

- For each journal, an issue with a given number is published in a single year.
- The `endpage` of an article is never smaller than the `startpage`.
- There is never (part of) more than one article on a single page.

The following is an instance of the relation:

ID	title	journal	issue	year	startpage	endpage	TR-ID
42	Cuckoo Hashing	JAlg	51	2004	121	133	87
33	Deterministic Dictionaries	JAlg	41	2001	69	85	62
33	Deterministic Dictionaries	JAlg	41	2001	69	85	56
39	Dictionaries in less space	SICOMP	31	2001	111	133	47
57	P vs NP resolved	JACM	51	2008	1	3	99
77	What Gödel missed	SICOMP	51	2008	1	5	98
78	What Gödel missed	Nature	2222	2008	22	22	98

a) Based on the above, indicate for each of the following sets of attributes whether it is a key for `Articles` or not. Use the answer sheet of the exam for your answer.

1. {ID};
2. {ID, TR-ID};
3. {ID, title, TR-ID}
4. {title};
5. {title, year};
6. {startpage, journal, issue}

If you wish, you may additionally write a brief explanation for each answer, which will be taken into account, but is not necessary to get full points.

Answer:

#	1	2	3	4	5	6
Key		X				
Not a key	X		X	X	X	X

b) Based on the above, indicate for each of the following potential functional dependencies, whether it is indeed an FD or not. Use the answer sheet of the exam for your answer.

1. `ID` \rightarrow `title`;
2. `startpage` \rightarrow `endpage`;
3. `journal issue` \rightarrow `year`
4. `title` \rightarrow `ID`;
5. `ID` \rightarrow `startpage endpage journal issue`;
6. `TR-ID` \rightarrow `ID`

If you wish, you may additionally write a brief explanation for each answer, which will be taken into account, but is not necessary to get full points.

Answer:

#	1	2	3	4	5	6
FD	X		X		X	
Not an FD		X		X		X

c) Based on a) and b), perform normalization into BCNF, and state the resulting relations.

Answer:

All FDs above are avoidable (by question a), so we decompose according to:

`journal issue → year`

`ID → title journal issue startpage endpage`

This gives the following relations:

`Articles1(journal,issue,year)`

`Articles2(ID,title,journal,issue,startpage,endpage)`

`Articles3(ID,TR-ID)`

3 SQL and relational algebra (35%)

We consider again the relation `Articles` from problem 2.

a) Indicate for each of the following expressions whether it is a valid SQL statement or not. A valid statement, as described in G UW, should be accepted by a standard SQL interpreter, whereas an invalid statement should result in an error message. Use the answer sheet of the exam for your answer.

1. `SELECT * FROM Articles WHERE endpage-startpage>10;`
2. `SELECT * FROM Articles WHERE endpage-startpage<0;`
3. `SELECT SUM(title) FROM Articles;`
4. `SELECT AVG(year) FROM Articles WHERE title LIKE 'C%';`
5. `SELECT COUNT(*) FROM Articles GROUP BY year;`
6. `SELECT year,COUNT(*) FROM Articles WHERE COUNT(*)>10 GROUP BY year;`

Answer:

#	1	2	3	4	5	6
Valid	X	X		X	X	
Invalid			X			X

b) Indicate for each of the following queries, how many tuples would be returned if it was run on the instance of `Articles` from problem 2. Use the answer sheet of the exam for your answer.

1. `SELECT ID FROM Articles WHERE year<2006;`
2. `SELECT DISTINCT ID FROM Articles WHERE year<2006;`
3. `SELECT AVG(year) FROM Articles GROUP BY journal;`
4. `SELECT ID FROM Articles WHERE title LIKE '%d';`

Answer:

#	1	2	3	4
Number of tuples	4	3	5	3

Consider the relations `Authors(auID,name)` and `Authoring(articleID,authorID)`, containing information on names of authors, and who is authoring which papers, respectively.

c) Write an SQL query that returns for each article, its ID, title and the number of authors.

Answer:

```
SELECT ID, title, COUNT(*)
FROM Articles, Authoring
WHERE ID=articleID
GROUP BY ID,title;
```

d) Write an SQL query that returns the titles of articles authored by 'Robert Tarjan'.

Answer:

```
SELECT title
FROM Articles, Authors, Authoring
WHERE ID=articleID AND auID=authorID AND name='Robert Tarjan';
```

e) Write an SQL query that returns the number of co-authors of 'Robert Tarjan'. (I.e., the number of authors who have written at least one article together with him.)

Answer:

```
SELECT COUNT(DISTINCT A2.authorID)
FROM Authors, Authoring A1, Authoring A2
WHERE A1.authorID=auID AND name='Robert Tarjan' AND
      A2.authorID<>auID AND A1.articleID=A2.articleID;
```

f) Write SQL statements that correspond to the following two relational algebra expressions. Duplicate elimination should be performed.

1. $\pi_{\text{title, year}}(\sigma_{\text{year}=2005}(\text{Articles}))$
2. $\gamma_{\text{year, COUNT(ID)}}(\text{Articles})$

Answer:

```
1. SELECT DISTINCT title, authorID FROM Articles WHERE
   year=2005;
2. SELECT year, COUNT(ID) FROM Articles GROUP BY year;
```

4 Efficiency and transactions (15%)

Consider the following six queries on `Articles` from problem 2:

1. `SELECT title FROM Articles WHERE year=2005;`
2. `SELECT title FROM Articles WHERE endpage=100;`
3. `SELECT title FROM Articles WHERE year>1995 AND year<2000;`
4. `SELECT title FROM Articles WHERE journal='JACM' AND issue=55;`
5. `SELECT title FROM Articles WHERE issue=55 AND journal='JACM';`
6. `SELECT title FROM Articles WHERE endpage-startpage>50;`

a) Indicate which of the above queries would likely be faster (based on the knowledge you have from the course), if *all* of the following indexes were created. Use the answer sheet of the exam for your answer.

```
CREATE INDEX Idx1 ON Articles(year,startpage);
CREATE INDEX Idx2 ON Articles(startpage,endpage);
CREATE INDEX Idx3 ON Articles(journal,issue,year);
```

Answer:

#	1	2	3	4	5	6
Faster	X		X	X	X	
Same		X				X

In the following we consider the below transactions on the `Authors(auID,name)` relation.

Time	User A	User B
1	<code>INSERT INTO Authors VALUES (42,'Donald Knuth');</code>	
2		<code>INSERT INTO Authors VALUES (43,'Guy Threepwood');</code>
3		<code>DELETE FROM Authors WHERE name LIKE 'Don%';</code>
4		<code>INSERT INTO Authors VALUES (44,'Donald E. Knuth');</code>
5	<code>DELETE FROM Authors WHERE name LIKE 'Guy%';</code>	
6	<code>COMMIT;</code>	
7		<code>COMMIT;</code>

b) Suppose that `Authors` is initially empty, that the transactions are run at isolation level `READ COMMITTED`, and that the commands are issued in the order indicated above. What is the content of `Authors` after the execution?

Answer:

auID	name
42	Donald Knuth
43	Guy Threepwood
44	Donald E. Knuth

c) Suppose that `Authors` is initially empty. What are the possible contents of `Authors` after each *serial* execution of the two transactions?

Answer:

auID	name
43	Guy Threepwood
44	Donald E. Knuth

auID	name
44	Donald E. Knuth
42	Donald Knuth

5 Constraints (10%)

Suppose that the `Authoring` relation of problem 3 relation was created as follows:

```
CREATE TABLE Authoring(  
    articleID INT REFERENCES Article(ID) ON DELETE SET NULL,  
    authorID INT REFERENCES Author(ID) ON DELETE CASCADE  
)
```

a) Indicate which of the following statements are true, and which are not. Use the answer sheet of the exam for your answer.

1. If we try to delete a tuple from `Authoring`, the tuple is not deleted. Instead, `articleID` is set to `NULL`.
2. If we delete a tuple from `Authoring`, any tuples in `Author` referred to by this tuple are also deleted.
3. If we delete a tuple from `Article`, some attributes of `Authoring` may have their values set to `NULL`.
4. If we try to insert a tuple into `Author`, with an `ID` that is not referred to in `Authoring`, the operation is rejected.
5. If we try to insert a tuple into `Authoring`, with an `ID` that does not exist in `Author`, the operation is rejected.

Answer:

#	1	2	3	4	5
True			X		X
False	X	X		X	

b) Write `CHECK` constraints for `Articles` of Problem 2 that ensure the following:

1. Values of the `journal` attribute does *not* start with 'Journal'.
2. The value of the `endpage` attribute is never smaller than that of `startpage`.
3. The value of `year` is given in full (e.g. 1999 is not abbreviated as 99). You may assume that `year` is of type integer, and that there are no articles more than 200 years old.

Answer:

1. `CHECK NOT (journal LIKE 'Journal%');`
2. `CHECK endpage >= startpage;`
3. `CHECK year >= 1000;`

Answer sheet (to be handed in)

Name		Page number	
CPR		Total pages	

Instructions. For all questions except 3.b (which asks for numbers), you must place exactly one X in each column. Note that the grading will be done in a way such that random answering does not pay. For example, two correct answers and one incorrect answer will be worth the same as one correct answer and two question marks.

Question 2.a	1	2	3	4	5	6
Key						
Not a key						
?						

Question 2.b	1	2	3	4	5	6
FD						
Not an FD						
?						

Question 3.a	1	2	3	4	5	6
Valid						
Invalid						
?						

Question 3.b	1	2	3	4
Number of tuples				

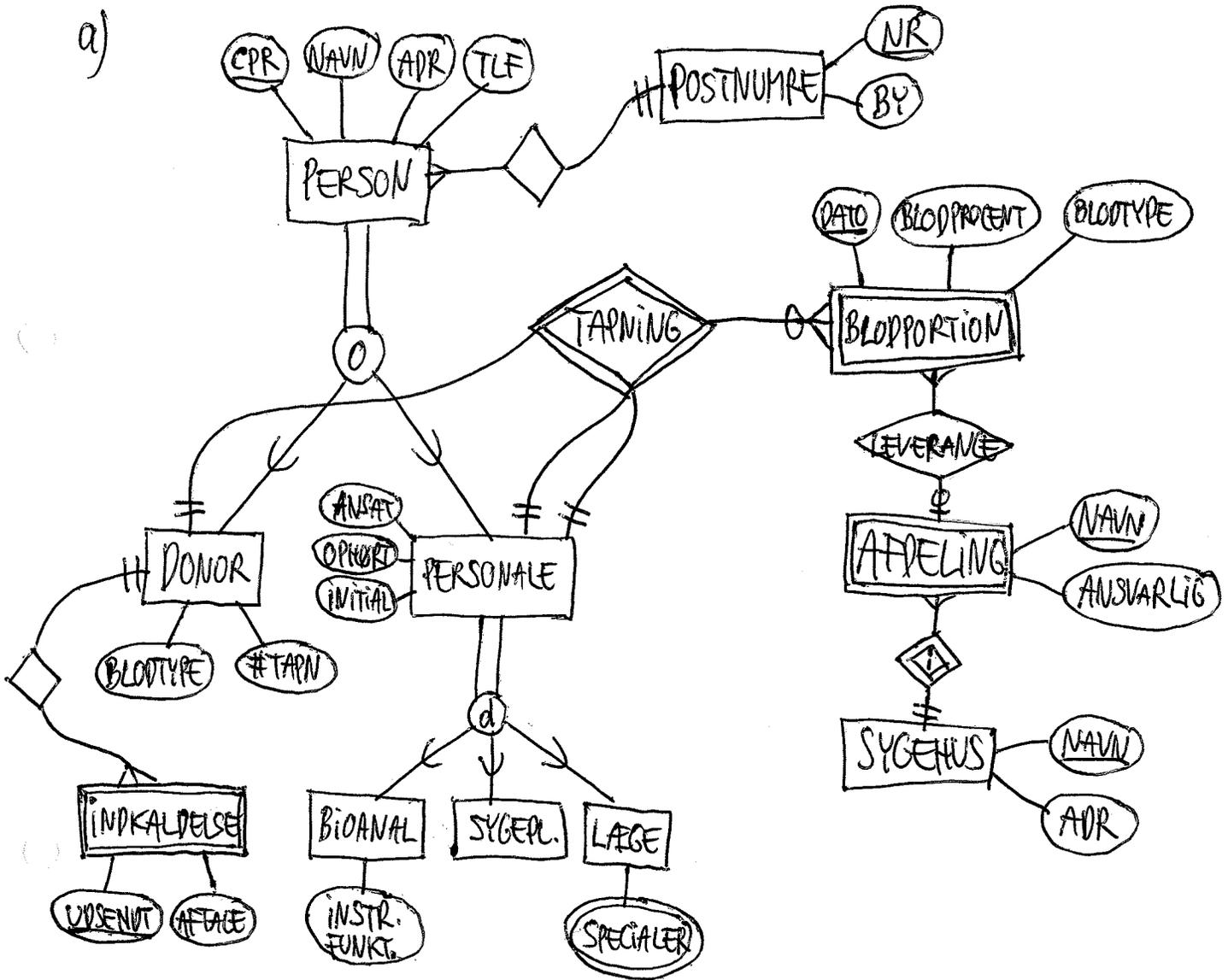
Question 4.a	1	2	3	4	5	6
Faster						
Same						
?						

Question 5.a	1	2	3	4	5
True					
False					
?					

DATABASESYSTEMER 7/6 - 2005

OPGAVE 1

a)



b) VED BRUG AF METODENE FRA FØRELÆSNINGSSLIDES:

PERSON (CPR, NAVN, ADR, TLF, POSTNR)
 POSTNUMRE (NR, BY)
 DONOR (CPR, BLODTYPE, #TAPN)
 PERSONALE (CPR, ANSAT, OPHØRT, INITIAL)
 INDKALDELSE (CPR, UDSENDT, AFTALE)
 BIOANAL (CPR, INSTR.FUNKT.)
 LÆGE (CPR, SPECIALER)

KAN EVT. INDEHOLDE "SUBTYPE DISCRIMINATOR"

BLODPORTION (DONORCPR, PERSONALECPR1, PERSONCPR2, DATO, BLODPROCENT, BLODTYPE, LEVERETTIL)
 AFDELING (NAVN, SYGEHUS, ANSVARLIG)
 SYGEHUS (NAVN, ADR)

OPGAVE 2

a) KANDIDATNØGLER: $\{id, sygdom\}$, $\{sygdom, by\}$, $\{sygdom, postnr\}$.

b) FLG. FUNKTIONELLE AFHÆNGIGHEDER HAR IKKE EN KANDIDATNØGLE PÅ VENSTE SIDE:

$id \rightarrow adresse, postnr, by$

$postnr \rightarrow by$

$sygdom \rightarrow speciale$

DA $postnr$ OG by BEGGE ER EN ÆGTE DELMÆNGDE AF EN KANDIDATNØGLE, GIVER KUN FØLGENDE $id \rightarrow adresse$ OG $sygdom \rightarrow speciale$ ANKENNING TIL DEKOMPOSITION:

$Adr(id, adresse)$

$Spec(sygdom, speciale)$

$Behandler2(id, postnr, by, sygdom)$

OPGAVE 3

SPECIALT SOM HUNDEGALSKAB HØRER UNDER FOR

a) 1 RETURNERER ALLE BEHANDLERE AF HUNDEGALSKAB I
POSTNR 1000 (IFØLGE OPG. 2 ER DER HØJST ÉN).

2 RETURNERER ALLE SPECIALER SOM DEN PÅGÆLDENDE
BEHANDLER HAR.

b) ^{distinct id}
SELECT speciale, COUNT(~~id~~), COUNT(DISTINCT sygdom)
FROM Behandler
GROUP BY speciale;

c) SELECT id
FROM Patient
WHERE sygdom NOT IN (SELECT sygdom
FROM Behandler
WHERE id = behandler_id);

d) CREATE TABLE Patient2 (
id INT, sygdom CHAR(32), behandler_by CHAR(16),
FOREIGN KEY (sygdom, behandler_by) REFERENCES Behandler(sygdom, by));

UPDATE Patient
SET ~~id~~ behandler_id = NULL
WHERE sygdom NOT IN (SELECT sygdom FROM Behandler
WHERE id = behandler_id);

} ÆNDRING AF
PATIENT KAN
UNDGÅS VED FØRST
AT LAVE EN KOPI

INSERT INTO Patient2 VALUES
(SELECT Patient.id, Patient.sygdom, Behandler.by
FROM Patient, Behandler
WHERE behandler_id = Behandler.id);

OPGAVE 4

a) PÅ ISOLERINGSNIVEAU READ COMMITTED KAN ÆNDRINGER LAVET AF ANDRE TRANSAKTIONER FØREKOMME MELLEM TO SQL SÆTNINGER.

TRANSAKTIONEN FINDER FRIE SÆVER I SÆTNING 3, OG ÉT AF DEM KAN SAGTENS BLIVE BOOKET INDEN SÆTNING 7.

b) PÅ "REPEATABLE READ" OG "SERIALIZABLE" KAN INGEN LÆST VÆRDI ÆNDRE SIG UNDER TRANSAKTIONEN, DERFOR KAN SEATS IKKE OPDATERES MELLEM SÆTNING 3 OG 7.

VED "SNAPSHOT ISOLATION" VIL TRANSAKTIONEN ABORTE HVIS DATA, DEN HAR SKREVET TIL, ER BLEVET ÆNDRET UNDERVEJS AF EN ANDEN TRANSAKTION.

DOBBELTBOOKING ER ALTSÅ UMULIG I ALLE 3 TILFÆLDE.

OPGAVE 5

a) INDEKS PÅ speciale HJÆLPER IKKE, DA speciale IKKE ER DEL AF NOGEN BETINGELSE.

INDEKS PÅ id HJÆLPER IKKE - id FOREKOMMER KUN I FORBINDELSE MED "IN" I BETINGELSEN. (EN SMART DBMS KUNNE MÅSKE UDNYTTE ET INDEKS PÅ id, HVIS SUBQUERYEN I 2 HAR ET LILLE RESULTAT).

INDEKS PÅ postnr ^{HEJTT} ELLER sygdom ER EN GOD IDE, DA BEGGE BLIVER BRUGT I SELEKTIVE SELECTS I 1 SÅVEL SOM 2.

- b)
- idx1 GØR BÅDE 1 OG 2 HURTIGERE. (SÆRLIGT 1)
 - idx2 GØR 1 OG SUBQUERYEN I 2 HURTIGERE (VVS. 2 LIDT HURTIGERE)
 - idx3 HJÆLPER IKKE PÅ 1, OG FORMENTLIG IKKE PÅ 2
(EN SMART DBMS KUNNE BRUGE idx3 VED 2 HVIS SUBQUERY HAR LILLE RES.)
 - idx4 HJÆLPER BÅDE I 1 OG 2

c)

FØRSTE INDEKS PÅ {sygdom, postnr} (idx 2)

ANDRE INDEKS PÅ {postnr}

↑
SELEKTIV ATTRIBUT I 2

↑
GØR 1 MEGET HURTIG. OG KAN BRUGES AF SUBQUERY I 2.

(EN SMART DBMS KUNNE UDNYTTE {postnr, id} HVIS SUBQUERYEN GIVER ET LILLE RESULTAT).

Introduction to Databases

IT University of Copenhagen

January 16, 2006

This exam consists of 5 problems with a total of 16 questions. The weight of each problem is stated. You have 4 hours to answer all 16 questions. The complete assignment consists of 12 numbered pages (including this page), *plus an answer sheet to be used for several of the questions.*

If you cannot give a complete answer to a question, try to give a partial answer. You may choose to write your answer in Danish or English. Write only on the front of sheets, and remember to write your CPR-number on each page. Please start your answer to each question at the top of a *new* page. Please order and number the pages before handing in.

G UW refers to *Database Systems – The Complete Book* by Hector Garcia-Molina, Jeff Ullman, and Jennifer Widom, 2002.

All written aids are allowed / Alle skriftlige hjælpemidler er tilladt.

1 Database design (25%)

The academic world is an interesting example of international cooperation and exchange. This problem is concerned with modeling of a database that contains information on researchers, academic institutions, and collaborations among researchers. A researcher can either be employed as a professor or a lab assistant. There are three kinds of professors: Assistant, associate, and full professors. The following should be stored:

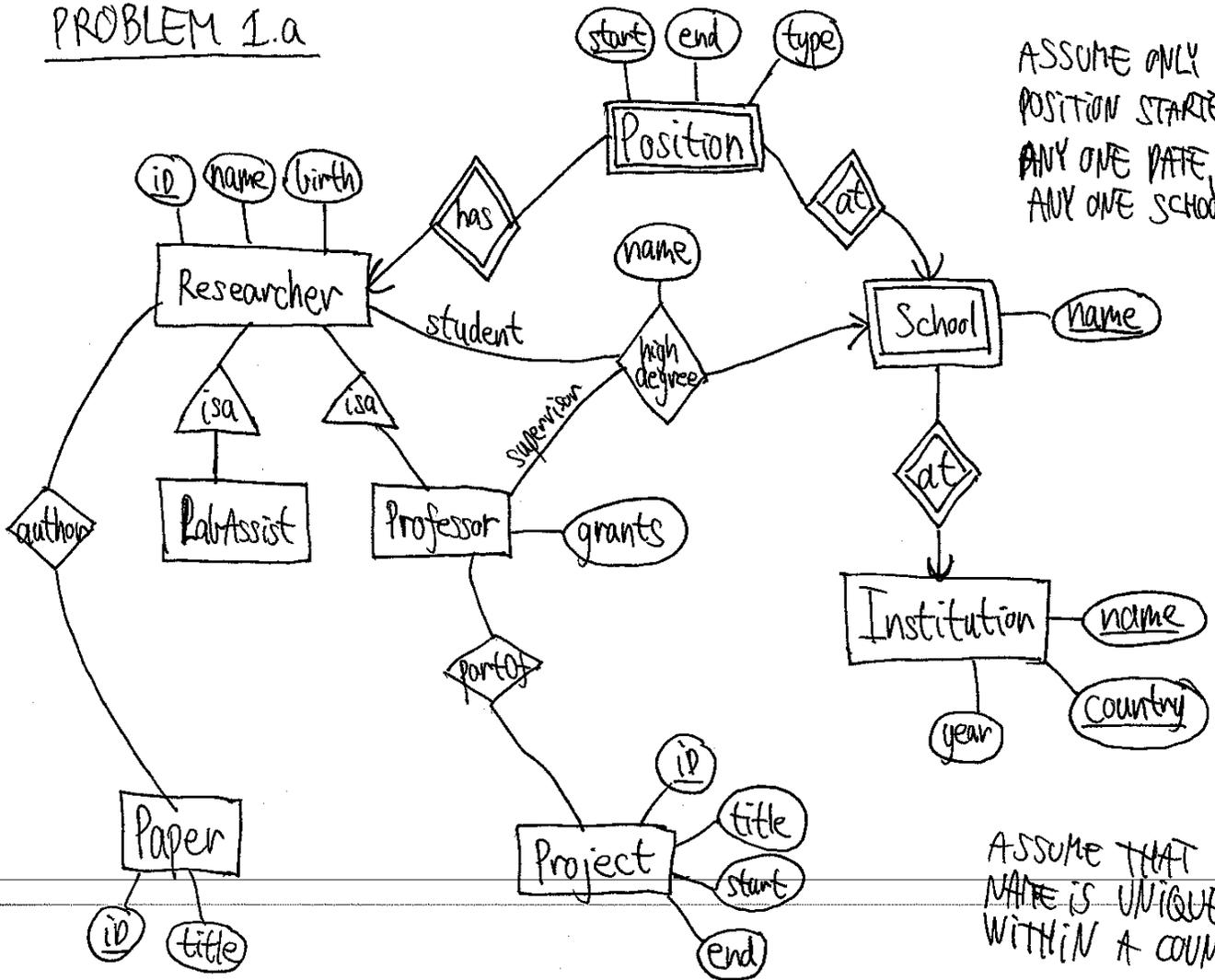
- For each researcher, his/her name, year of birth, and current position (if any).
- For each institution, its name, country, and inauguration year.
- For each institution, the names of its schools (*e.g. School of Law, School of Business, School of Computer Science,...*). A school belongs to exactly one institution.
- An employment history, including information on all employments (start and end date, position, and what school).
- Information about co-authorships, i.e., which researchers have co-authored a research paper. The titles of common research papers should also be stored.
- For each researcher, information on his/her highest degree (BSc, MSc or PhD), including who was the main supervisor, and at what school.
- For each professor, information on what research projects (title, start date, and end date) he/she is involved in, and the total amount of grant money for which he/she was the main applicant.

a) Draw an E/R diagram for the data set described above. Make sure to indicate all cardinality constraints specified above. The E/R diagram should not contain redundant entity sets, relationships, or attributes. Also, use relationships whenever appropriate. If you need to make any assumptions, include them in your answer.

Answer:

See next page.

PROBLEM 1.a



ASSUME ONLY ONE POSITION STARTED AT ANY ONE DATE AT ANY ONE SCHOOL

ASSUME THAT NAME IS UNIQUE WITHIN A COUNTRY

b) Convert your E/R diagram from question a) into relations, and write SQL statements to create the relations. You may make any reasonable choice of data types. Remember to include any constraints that follow from the description of the data set or your E/R diagram, including primary key and foreign key constraints.

Answer:

```
CREATE TABLE Researcher (  
  ID int PRIMARY KEY,  
  name VARCHAR(50),  
  birth INT  
);
```

```
CREATE TABLE Professor (  
  ID int REFERENCES Researcher(ID),  
  grants INT  
);
```

```
CREATE TABLE Paper (  
  ID int PRIMARY KEY,  
  title VARCHAR(50)  
);
```

```
CREATE TABLE Position (  
  researcherID int REFERENCES Researcher(ID),  
  start DATE,  
  end DATE,  
  schoolName VARCHAR(50),  
  institutionName VARCHAR(50),  
  FOREIGN KEY (schoolName,institutionName) REFERENCES School(name,insitutionName),  
  PRIMARY KEY (researcherID,start,schoolName,institutionName)  
);
```

```
CREATE TABLE Project (  
  ID int PRIMARY KEY,  
  title VARCHAR(50),  
  start DATE,  
  end DATE  
);
```

```
CREATE TABLE School (  
  name VARCHAR(50),  
  institutionName VARCHAR(50) REFERENCES Institution(name),  
  PRIMARY KEY (name, institutionName)  
);
```

```
CREATE TABLE Institution (  
name VARCHAR(50),  
country VARCHAR(50),  
year INT,  
PRIMARY KEY (name, country)  
);
```

```
CREATE TABLE Author (  
researcherID INT REFERENCES Researcher(ID),  
paperID INT REFERENCES Paper(ID),  
PRIMARY KEY (researcherID, paperID)  
);
```

```
CREATE TABLE PartOf (  
professorID INT REFERENCES Professor(ID),  
studentID INT REFERENCES Researcher(ID),  
PRIMARY KEY (professorID, projectID)  
);
```

```
CREATE TABLE HighDegree (  
professorID INT REFERENCES Professor(ID),  
projectID INT REFERENCES Project(ID),  
schoolName VARCHAR(50),  
institutionName VARCHAR(50),  
FOREIGN KEY (schoolName,institutionName) REFERENCES School(name,insitutionName),  
PRIMARY KEY (professorID, studentID, schoolName, institutionName)  
);
```

2 Normalization (15%)

We consider the following relation:

`Articles(ID, title, journal, issue, year, startpage, endpage, TR-ID)`

It contains information on articles published in scientific journals. Each article has a unique ID, a title, and information on where to find it (name of journal, what issue, and on which pages). Also, if results of an article previously appeared in a “technical report” (TR), the ID of this technical report can be specified. We have the following information on the attributes:

- For each journal, an issue with a given number is published in a single year.
- The `endpage` of an article is never smaller than the `startpage`.
- There is never (part of) more than one article on a single page.

The following is an instance of the relation:

ID	title	journal	issue	year	startpage	endpage	TR-ID
42	Cuckoo Hashing	JAlg	51	2004	121	133	87
33	Deterministic Dictionaries	JAlg	41	2001	69	85	62
33	Deterministic Dictionaries	JAlg	41	2001	69	85	56
39	Dictionaries in less space	SICOMP	31	2001	111	133	47
57	P vs NP resolved	JACM	51	2008	1	3	99
77	What Gödel missed	SICOMP	51	2008	1	5	98
78	What Gödel missed	Nature	2222	2008	22	22	98

a) Based on the above, indicate for each of the following sets of attributes whether it is a key for `Articles` or not. Use the answer sheet of the exam for your answer.

1. {ID};
2. {ID, TR-ID};
3. {ID, title, TR-ID}
4. {title};
5. {title, year};
6. {startpage, journal, issue}

If you wish, you may additionally write a brief explanation for each answer, which will be taken into account, but is not necessary to get full points.

Answer:

#	1	2	3	4	5	6
Key		X				
Not a key	X		X	X	X	X

b) Based on the above, indicate for each of the following potential functional dependencies, whether it is indeed an FD or not. Use the answer sheet of the exam for your answer.

1. `ID` \rightarrow `title`;
2. `startpage` \rightarrow `endpage`;
3. `journal issue` \rightarrow `year`
4. `title` \rightarrow `ID`;
5. `ID` \rightarrow `startpage` `endpage` `journal` `issue`;
6. `TR-ID` \rightarrow `ID`

If you wish, you may additionally write a brief explanation for each answer, which will be taken into account, but is not necessary to get full points.

Answer:

#	1	2	3	4	5	6
FD	X		X		X	
Not an FD		X		X		X

c) Based on a) and b), perform normalization into BCNF, and state the resulting relations.

Answer:

All FDs above are avoidable (by question a), so we decompose according to:

`journal issue → year`

`ID → title journal issue startpage endpage`

This gives the following relations:

`Articles1(journal,issue,year)`

`Articles2(ID,title,journal,issue,startpage,endpage)`

`Articles3(ID,TR-ID)`

3 SQL and relational algebra (35%)

We consider again the relation `Articles` from problem 2.

a) Indicate for each of the following expressions whether it is a valid SQL statement or not. A valid statement, as described in G UW, should be accepted by a standard SQL interpreter, whereas an invalid statement should result in an error message. Use the answer sheet of the exam for your answer.

1. `SELECT * FROM Articles WHERE endpage-startpage>10;`
2. `SELECT * FROM Articles WHERE endpage-startpage<0;`
3. `SELECT SUM(title) FROM Articles;`
4. `SELECT AVG(year) FROM Articles WHERE title LIKE 'C%';`
5. `SELECT COUNT(*) FROM Articles GROUP BY year;`
6. `SELECT year,COUNT(*) FROM Articles WHERE COUNT(*)>10 GROUP BY year;`

Answer:

#	1	2	3	4	5	6
Valid	X	X		X	X	
Invalid			X			X

b) Indicate for each of the following queries, how many tuples would be returned if it was run on the instance of `Articles` from problem 2. Use the answer sheet of the exam for your answer.

1. `SELECT ID FROM Articles WHERE year<2006;`
2. `SELECT DISTINCT ID FROM Articles WHERE year<2006;`
3. `SELECT AVG(year) FROM Articles GROUP BY journal;`
4. `SELECT ID FROM Articles WHERE title LIKE '%d';`

Answer:

#	1	2	3	4
Number of tuples	4	3	5	3

Consider the relations `Authors(auID,name)` and `Authoring(articleID,authorID)`, containing information on names of authors, and who is authoring which papers, respectively.

c) Write an SQL query that returns for each article, its ID, title and the number of authors.

Answer:

```
SELECT ID, title, COUNT(*)
FROM Articles, Authoring
WHERE ID=articleID
GROUP BY ID,title;
```

d) Write an SQL query that returns the titles of articles authored by 'Robert Tarjan'.

Answer:

```
SELECT title
FROM Articles, Authors, Authoring
WHERE ID=articleID AND auID=authorID AND name='Robert Tarjan';
```

e) Write an SQL query that returns the number of co-authors of 'Robert Tarjan'. (I.e., the number of authors who have written at least one article together with him.)

Answer:

```
SELECT COUNT(DISTINCT A2.authorID)
FROM Authors, Authoring A1, Authoring A2
WHERE A1.authorID=auID AND name='Robert Tarjan' AND
      A2.authorID<>auID AND A1.articleID=A2.articleID;
```

f) Write SQL statements that correspond to the following two relational algebra expressions. Duplicate elimination should be performed.

1. $\pi_{\text{title, year}}(\sigma_{\text{year}=2005}(\text{Articles}))$
2. $\gamma_{\text{year, COUNT(ID)}}(\text{Articles})$

Answer:

```
1. SELECT DISTINCT title, authorID FROM Articles WHERE
   year=2005;
2. SELECT year, COUNT(ID) FROM Articles GROUP BY year;
```

4 Efficiency and transactions (15%)

Consider the following six queries on `Articles` from problem 2:

1. `SELECT title FROM Articles WHERE year=2005;`
2. `SELECT title FROM Articles WHERE endpage=100;`
3. `SELECT title FROM Articles WHERE year>1995 AND year<2000;`
4. `SELECT title FROM Articles WHERE journal='JACM' AND issue=55;`
5. `SELECT title FROM Articles WHERE issue=55 AND journal='JACM';`
6. `SELECT title FROM Articles WHERE endpage-startpage>50;`

a) Indicate which of the above queries would likely be faster (based on the knowledge you have from the course), if *all* of the following indexes were created. Use the answer sheet of the exam for your answer.

```
CREATE INDEX Idx1 ON Articles(year,startpage);
CREATE INDEX Idx2 ON Articles(startpage,endpage);
CREATE INDEX Idx3 ON Articles(journal,issue,year);
```

Answer:

#	1	2	3	4	5	6
Faster	X		X	X	X	
Same		X				X

In the following we consider the below transactions on the `Authors(auID,name)` relation.

Time	User A	User B
1	<code>INSERT INTO Authors VALUES (42,'Donald Knuth');</code>	
2		<code>INSERT INTO Authors VALUES (43,'Guy Threepwood');</code>
3		<code>DELETE FROM Authors WHERE name LIKE 'Don%';</code>
4		<code>INSERT INTO Authors VALUES (44,'Donald E. Knuth');</code>
5	<code>DELETE FROM Authors WHERE name LIKE 'Guy%';</code>	
6	<code>COMMIT;</code>	
7		<code>COMMIT;</code>

b) Suppose that `Authors` is initially empty, that the transactions are run at isolation level `READ COMMITTED`, and that the commands are issued in the order indicated above. What is the content of `Authors` after the execution?

Answer:

auID	name
42	Donald Knuth
43	Guy Threepwood
44	Donald E. Knuth

c) Suppose that `Authors` is initially empty. What are the possible contents of `Authors` after each *serial* execution of the two transactions?

Answer:

auID	name
43	Guy Threepwood
44	Donald E. Knuth

auID	name
44	Donald E. Knuth
42	Donald Knuth

5 Constraints (10%)

Suppose that the `Authoring` relation of problem 3 relation was created as follows:

```
CREATE TABLE Authoring(  
    articleID INT REFERENCES Article(ID) ON DELETE SET NULL,  
    authorID INT REFERENCES Author(ID) ON DELETE CASCADE  
)
```

a) Indicate which of the following statements are true, and which are not. Use the answer sheet of the exam for your answer.

1. If we try to delete a tuple from `Authoring`, the tuple is not deleted. Instead, `articleID` is set to `NULL`.
2. If we delete a tuple from `Authoring`, any tuples in `Author` referred to by this tuple are also deleted.
3. If we delete a tuple from `Article`, some attributes of `Authoring` may have their values set to `NULL`.
4. If we try to insert a tuple into `Author`, with an `ID` that is not referred to in `Authoring`, the operation is rejected.
5. If we try to insert a tuple into `Authoring`, with an `ID` that does not exist in `Author`, the operation is rejected.

Answer:

#	1	2	3	4	5
True			X		X
False	X	X		X	

b) Write `CHECK` constraints for `Articles` of Problem 2 that ensure the following:

1. Values of the `journal` attribute does *not* start with 'Journal'.
2. The value of the `endpage` attribute is never smaller than that of `startpage`.
3. The value of `year` is given in full (e.g. 1999 is not abbreviated as 99). You may assume that `year` is of type integer, and that there are no articles more than 200 years old.

Answer:

1. `CHECK NOT (journal LIKE 'Journal%');`
2. `CHECK endpage>=startpage;`
3. `CHECK year>=1000;`

Answer sheet (to be handed in)

Name		Page number	
CPR		Total pages	

Instructions. For all questions except 3.b (which asks for numbers), you must place exactly one X in each column. Note that the grading will be done in a way such that random answering does not pay. For example, two correct answers and one incorrect answer will be worth the same as one correct answer and two question marks.

Question 2.a	1	2	3	4	5	6
Key						
Not a key						
?						

Question 2.b	1	2	3	4	5	6
FD						
Not an FD						
?						

Question 3.a	1	2	3	4	5	6
Valid						
Invalid						
?						

Question 3.b	1	2	3	4
Number of tuples				

Question 4.a	1	2	3	4	5	6
Faster						
Same						
?						

Question 5.a	1	2	3	4	5
True					
False					
?					