NUNStA Online Book Exchange

Students at the National University of Ngendipura (NUN) buy books for their studies. They also lend and borrow books from other students. Your company, Apasaja Pt. Ltd., is commissioned by NUN Students Association (NUNStA) to design and implement an online book exchange system for its students.

Apasaja Pt. Ltd. designs and implements a database application that records information about students, books they own and books they borrow from other students.

The database records the name, faculty, department and student number of each student. Each student is identified in the system by its email. The database also records the date at which the student joined the university.

The database records the title, authors, publisher, year and edition and the ISBN-10 and ISBN-13 for each book. The International Standard Book Number, ISBN-10 or -13, is an industry standard for the unique identification of books. It is possible that the database records books that are not owned by any students (because the owners of a copy graduated or because the book was advised by a lecturer for a course but not yet purchased by any student.)

The database records the date at which a book copy is borrowed and the date at which it is returned. We will refer to this information as a loan record.

For historical purposes the database records information about the copies and the owners of copies as long as the owners are students or there are loan records concerning the copies.

For historical purposes the database records information about graduated students as long as there are loan records concerning books that they owned.

For historical purposes (in order to keep the loan records for this book) the database records the case of a book that was owned and subsequently sold unless the copy was never borrowed.

```
CREATE TABLE book (
title VARCHAR(256) NOT NULL,
format CHAR(9) CHECK(format = 'paperback' OR format='hardcover'),
pages INT,
language VARCHAR(32),
authors VARCHAR(256),
publisher VARCHAR(64),
year DATE,
ISBN10 CHAR(10) NOT NULL UNIQUE,
ISBN13 CHAR(14) PRIMARY KEY
)

CREATE TABLE student (
name VARCHAR(32) NOT NULL,
email VARCHAR(256) PRIMARY KEY,
year DATE NOT NULL,
faculty VARCHAR(62) NOT NULL,
department VARCHAR(32) NOT NULL,
graduate DATE,
CHECK(graduate >= year)
)

CREATE TABLE copy (
owner VARCHAR(256) REFERENCES student(email) ON UPDATE CASCADE ON DELETE CASCADE,
book CHAR(14) REFERENCES book(ISBN13) ON UPDATE CASCADE,
copy INT CHECK(copy>0),
available BIT NOT NULL DEFAULT 'TRUE',
PRIMARY KEY (owner, book, copy)
)

CREATE TABLE loan (
borrower VARCHAR(256) REFERENCES student(email),
owner VARCHAR(256)
book CHAR(14),
copy INT,
borrowed DATE,
returned DATE,
FOREIGN KEY (owner, book, copy) REFERENCES copy(owner, book, copy) ON UPDATE CASCADE ON
DELETE CASCADE,
PRIMARY KEY (borrowed, borrower, owner, book, copy),
CHECK(returned >= borrowed)
)
```

Tutorial 1

1) Create the table that contains the following information about books: title, format (paperback or hardcover), number of pages, authors, publisher, year, edition, ISBN-10 and -13. Check out one book on the Web, for instance on amazon.com, to see some examples of the values of these attributes. Check out the Web for the available SQL domains (types) and try them with your database management system. Choose a primary key. Forbid NULL values.

   ```
   CREATE TABLE book (
   title VARCHAR(128) NOT NULL,
   format CHAR(9) CHECK(format = 'paperback' OR format='hardcover' OR format IS NULL),
   pages INT,
   authors VARCHAR(128),
   publisher VARCHAR(32),
   year DATE,
   ISBN10 CHAR(10) NOT NULL UNIQUE,
   ISBN13 CHAR(14) PRIMARY KEY,
   )
   ```

2) Print all the information about books.

   ```
   SELECT * FROM book
   ```

3) Delete the relation book.

   ```
   DROP TABLE book
   ```

4) Re-create the table books.

   ```
   CREATE TABLE book (
   title VARCHAR(128) NOT NULL,
   format CHAR(9) CHECK(format = 'paperback' OR format='hardcover' OR format IS NULL),
   pages INT,
   authors VARCHAR(128),
   publisher VARCHAR(32),
   year DATE,
   ISBN10 CHAR(10) NOT NULL UNIQUE,
   ISBN13 CHAR(14) PRIMARY KEY,
   )
   ```

5) Insert one book called "Introduction to Database Systems". Go to the Web to find actual details.

   ```
   INSERT INTO book VALUES ('Introduction to Database Systems', 'Paperback', 168, 'Stephane Bressan and Barbara Catania', 'MacGraw-Hill', '2005-01-01', '0071246509', '978-0071246507')
   ```

6) Insert half a dozen books you can find with title containing "Introduction to Database Systems" or authored by C.J. Date. Go to the Web, for instance amazon.com, to find the details.

   ```
   INSERT INTO book VALUES ('An Introduction to Database Systems', 'Paperback', 1024, 'C.J. Date', 'Addison-Wesley',  '2003-08-01', '0321197844', '978-0321197849')
   ```

INSERT INTO book VALUES ('Introduction to Database Systems (Management Information Systems)', 'Paperback', 29, 'R. Dixon, G. Rawlings', 'CIMA Publishing', '1998-12-01', '0948036230', '978-0948036231')

INSERT INTO book VALUES ('Introduction to Database Systems', 'Hardcover', 650, 'Bipin C. Desai', 'West Group', '1990-08-01', '0314667717', '978-0314667717')

INSERT INTO book VALUES ('SQL and Relational Theory: How to Write Accurate SQL Code', 'Paperback', 432, 'C.J. Date', 'O Reilly Media', '2009-01-23, '0596523068', '978-0596523060')

7) Modify all the Books authored by C.J. Date to mention the author's first name (find the author's first name from the Web.)

   UPDATE book SET authors='Christopher J. Date' WHERE authors= 'C. J. Date'

8) Print all the information about books

   SELECT * FROM book

9) Delete all the books authored by C.J. Date

   DELETE FROM book
   WHERE authors='Christopher J. Date'

   DELETE FROM book
   WHERE authors LIKE 'C%Date'

10) Find the title, format, number of pages, authors, publisher, year, edition, ISBN-10 and -13 of the books.

    SELECT title, format, pages, authors, publisher, year, edition, ISBN10, ISBN13 FROM book

11) Find the titles of the books.

    SELECT title FROM book

    SELECT DISTINCT title FROM book

12) Find the authors of the books called "Introduction to Database Systems".

    SELECT authors FROM book WHERE title='Introduction to Database Systems'

13) Add a language attribute to all books. Set the default language to English.

    ALTER TABLE book
    ADD language VARCHAR(32) DEFAULT 'English'

INSERT INTO book (title, format, pages, authors, publisher, year, ISBN10, ISBN13) VALUES ('SQL and Relational Theory: How to Write Accurate SQL Code', 'Paperback', 432, 'C.J. Date', 'O Reilly Media', '2009-01-23', '0596523068', '978-0596523060')

14) Create the tables for the remainder of this tutorial using the code in NUSStASchema.sql. Populate the tables using NUSStAData.sql. You can use NUSStAClean.sql to remove all data and tables. The files are available in IVLE workbin.

Tutorial 2 queries, duplicates, referential integrity, multiple relations and conditions and complex conditions, UNION.

Always ask yourself the question about duplicates potentially created by the query with or without DISTINCT. Only use DISTINCT when strictly necessary.

1) Create the tables using the file NUNStASchema.sql.

   Rearrange the statements

2) Populate the database using the files NUMNStABook.sql, NUMNStACopy.sql, NUMNStALoan.sql and NUMNStAStudent.sql.

   Check the foreign key constraints to decide in which order to insert the data.

3) You can clean up data and tables anytime using the file NUNStAClean.sql.

4) Find the different emails of students

   ```
   SELECT s.email
   FROM student s
   ```

5) Find the different emails of students

   ```
   SELECT s.email
   FROM student s
   ```

   DISTINCT is not needed

6) Print the names of students in descending alphabetical order.

   ```
   SELECT s.name
   FROM student s
   ORDER BY name DESC
   ```

7) Are there students with the same name?

   ```
   SELECT * FROM student s1, student s2 WHERE s1.name=s2.name AND s1.email < s2.email
   ```

8) Find the different names of students. Is the result sorted?

   ```
   SELECT DISTINCT s.name
   FROM student s
   ```

   The sorted result is a side effect of the duplicate elimination possibly specific to the DBMS. It is not guaranteed.

9) Find the names of students who owned a copy of book ‘978-0262033848’.

```
SELECT s.name
FROM student s, copy c
WHERE c.owner=s.email AND c.book='978-0262033848'
```

Do not use table book as it is guaranteed by referential integrity that there is such a book if it appears in table copy.

10) Find the names of students who owned a copy of book whose title contains 'computer' with more than 100 pages.

```
SELECT s.name
FROM student s, copy c, book b
WHERE c.owner=s.email AND c.book=b.ISBN13 AND b.title LIKE '%computer%' AND b.pages > 100
```

11) Find the number of A4 pages needed to photocopy the two books with ISBN-13 '978-0262033848' and '978-0321295354' (if you own the books and for personal use only!)

```
SELECT (b1.pages + b2.pages)/2
FROM book b1, book b2
WHERE b1.ISBN13 = '978-0262033848' AND b2.ISBN13='978-0321295354'
```

12) Find the different names of students who owned a copy of book other that '978-0262033848'.

```
SELECT DISTINCT s.name
FROM student s, copy c
WHERE c.owner=s.email AND c.book <> '978-0262033848'
```

This is not "Print the names of the different students …" why?

13) What is the meaning of the following query? SELECT s.name FROM student s, copy c WHERE NOT(c.owner=s.email AND c.book <> '978-0262033848')

```
SELECT DISTINCT s.name
FROM student s, copy c
WHERE c.owner<>s.email OR c.book <> '978-0262033848'
```

This prints all the students!

14) Find the names of students who borrowed a copy of book '978-0262033848'.

```
SELECT s.name
FROM student s, loan l
WHERE l.borrower=s.email AND l.book='978-0262033848'
```

15) Find the names of students who owned or borrowed a copy of book '978-0262033848'. Use UNION.

```
SELECT s.name
FROM student s, copy c
WHERE c.owner=s.email AND c.book='978-0262033848'
UNION
SELECT s.name
FROM student s, loan l
WHERE l.borrower=s.email AND l.book='978-0262033848'
```

16) Find the names of students who owned or borrowed a copy of book '978-0262033848'. USE OR.

```
SELECT s.name
FROM student s, copy c, loan l
WHERE (c.owner=s.email AND c.book='978-0262033848')
OR (l.borrower=s.email AND l.book='978-0262033848')
```

17) Delete all the data in table copy

```
DELETE FROM copy
```

18) Try again queries (12) and (13)

We see that the query with OR returns no results. This is wrong. That means that the query was not correct. It happens because the Cartesian product in the FROM clause is empty. When is it correct to use OR? OR should be used among conditions on the exactly same tables.

Tutorial 3 Aggregate functions, nested queries, views and triggers.

1) Find the number of copies in the system.

```
SELECT COUNT(*)
FROM copy c
```

2) Find, for each book, the corresponding number of copies. [Print the primary key of the book and the number of copies.]

```
SELECT c.book, COUNT(*)
FROM copy c
GROUP BY c.book
```

3) Find the available books with the largest number of copies.

```
SELECT c.book
FROM copy c
WHERE c.available='TRUE'
GROUP BY c.book
HAVING COUNT(*) >= ALL (SELECT COUNT(*) FROM copy c WHERE c.available='TRUE' GROUP BY c.book)
```

4) Find the names of the students that have borrowed some book by ‘Charles Dickens.

```
SELECT s.name
FROM student s, loan l, book b
WHERE l.borrower=s.email AND l.book=b.ISBN13 AND b.authors='Charles Dickens'
```

5) Find the number of different books by ‘Charles Dickens.

```
SELECT COUNT(DISTINCT s.email)
FROM student s, loan l, book b
WHERE l.borrower=s.email AND l.book=b.ISBN13 AND b.authors='Charles Dickens'
```

Note that we do not need the copy table by transitivity of the referential integrity.

6) Find the names of the different students that have borrowed all the books by ‘Charles Dickens’. Use aggregate functions.

```
SELECT s.name
FROM student s, loan l, book b
WHERE l.borrower=s.email AND l.book=b.ISBN13 AND b.authors='Charles Dickens'
GROUP BY s.name, s.email
HAVING COUNT(DISTINCT b.ISBN13) = (SELECT COUNT(*)
        FROM book b
        WHERE b.authors='Charles Dickens')
```

7) Find the names of students who owned a copy of book whose title contains ‘computer’ with more than 100 pages. Use nested queries. This is not the preferred answer.

```
SELECT s.name
FROM student s, copy c
WHERE c.owner=s.email AND c.book IN (
SELECT b.ISBN13 FROM book b WHERE b.title LIKE '%computer%' AND b.pages > 100)
```

```
SELECT s.name
FROM student s
WHERE s.email IN (
SELECT c.owner FROM copy c WHERE c.book IN (
SELECT b.ISBN13 FROM book b WHERE b.title LIKE '%computer%' AND b.pages > 100))
```

```
SELECT s.name
FROM student s, copy c, book b
WHERE s.email = c.owner AND c.book = b.ISBN13 AND b.title LIKE '%computer%' AND b.pages
<100
```

8) Find the different names of students who never owned a copy of book other that ‘978-0262033848’.

```
SELECT DISTINCT s.name
FROM student s
WHERE s.email NOT IN (SELECT c.owner FROM copy c WHERE  c.book = '978-0470128725')
```

9) Find the names of the different students that have borrowed all the books by ‘Charles Dickens’. Use NOT EXISTS. (You may also try with NOT IN and EXCEPT.)

```
SELECT s.name
FROM student s
WHERE NOT EXISTS
(SELECT *
FROM book b
WHERE b.authors=‘Charles Dickens’ AND NOT EXISTS
(SELECT *
FROM loan l
WHERE l.book=b.ISBN13 AND l.borrower=s.email))
```

10) Find the names of the different students that have borrowed all the books by ‘Amelie Nothomb’.

There is no such book so … every student has borrowed all her books! Or none?

```
SELECT s.name
FROM student s, loan l, book b
WHERE l.borrower=s.email AND l.book=b.ISBN13 AND b.authors='Amelie Nothomb'
```

```
GROUP BY s.name, s.email
HAVING COUNT(DISTINCT b.ISBN13) = (SELECT COUNT(*)
FROM book b
WHERE b.authors='Amelie Nothomb')


SELECT s.name
FROM student s
WHERE NOT EXISTS
(SELECT *
FROM book b
WHERE b.authors='Amelie Nothomb' AND NOT EXISTS
(SELECT *
FROM loan l
WHERE l.book=b.ISBN13 AND l.borrower=s.email))
```