CSE 411 / L-17 / 28.10.2024 /

Midterm Exam

- No content

L-18 / 30.10.2024 /

Two task for Database Recovery:

- writing log records during normal running period.
- if failure occures, recover from failure using the log records.

A = 1000
B = 2000

Transaction (T_0)

```
READ (A)
A = A - 100
WRITE (A)
READ (B)
B = B + 100
WRITE (B)
COMMIT
```

Log

$\langle T_0, START \rangle$

$\langle T_0, A, 1000, 900 \rangle$

$\langle T_0, B, 2000, 2100 \rangle$

$\langle T_0, COMMIT \rangle$

- if failed here, no need to undo, need to do redo.

if failed here, need to undo the transaction.

Compensation Log Records

- UNDO (T_0)
- $\langle T_0, B, 2000 \rangle$
- $\langle T_0, A, 1000 \rangle$
- $\langle T_0, Abort \rangle$

- UNDO (T): writes the old value

  REDO (T) : writes the new value

- Failure can also occure when the system is recoverying from another failure.

- UNDO (T):
  - → ⟨Ti start⟩ exist but,
  - → ⟨Ti commit⟩ or ⟨Ti Abort⟩ doesn't exist

- REDO (T):
  - Contains ⟨start⟩, ⟨commit⟩, ⟨abort⟩

- Repeating history:
  - REDO redoes all the original actions including the steps that restored at old value.

Slide - 41

- Checkpoint:
  - streamline recovery procedure
  - all updates are stopped during checkpoint.
  - output all records / modified buffer blocks to the disk.
  - write log records ⟨checkpoint L⟩
    - L → list of active transaction.
  - scan backward.

$\langle T_2, start \rangle$

$\langle checkpoint\ T_2 \rangle$ → all commit before this will be ignored.

$\langle T_2, commit \rangle$ → T_2 will be REDO, as it was committed after checkpoint.

$\langle T_3, start \rangle$

Example - Slide - 44

Check-Point is important for Exam

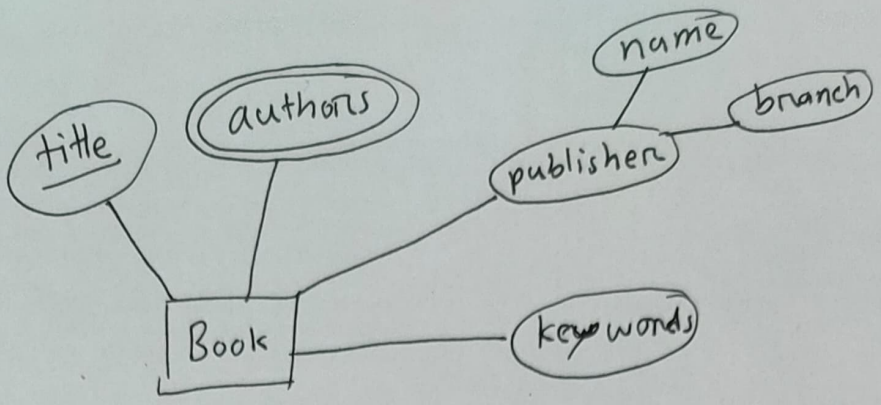
Quiz-2
upto this
13.11.2024

L-19 / 04.11.2024 /

Object Relational Data Models

- deal with added data types
- complex type, non-atomic values. - nested relations.
  - set of integers
  - set of tuples
- allow relation whenever we allow atomic (scalar) values.
  - relations within relations.
- violates first normal form (1NF).

Example - Slide - 5



⇒ 1NF Relation:

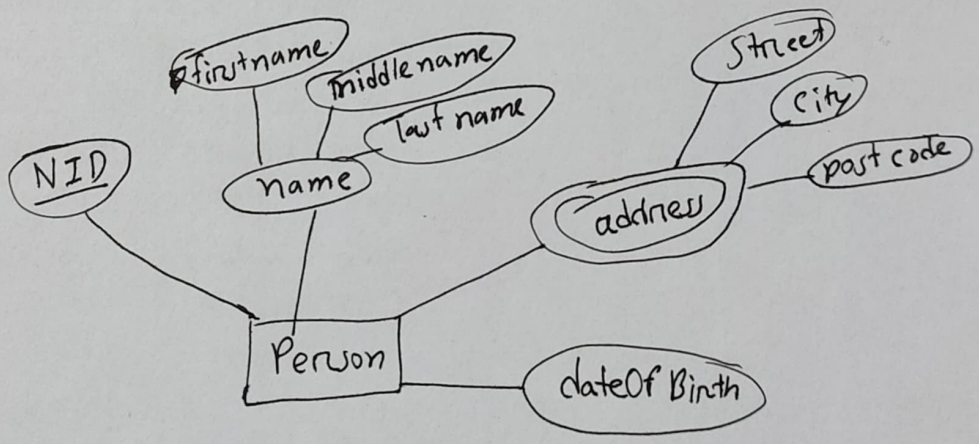
Book(title, publisher-name, publisher-branch)

Book-authors (title, author)

Book-keywords (title, keywords)

In object Relational Data models:

- we use array and set
  - array → ordered
  - set → non-ordered.



Create Table Person(NID varchar(10),

~~Name name~~

name Name,

address Address[3],

dateOfBirth date)

⇒ Insert Into Persons Values (

"12345789",

Function of datatype { new Name ("Joy", "Kumar", "Ghosh"),
new Address[0] ("Bashundhara", "Dhaka", 1229),
"2000-...-...");

- Reduced the number of table for multivalued field.

- Complex Types and SQL:

- structured types / user-defined types:

create type Name as (

firstname varchar(20),
lastname varchar(39,)

final,

⇒ final ⇒ no subtypes
not final ⇒ subtypes allowed.

- user-defined row types:

create type CustomerType as (

name Name,
address Address,
dateOfBirth date) not final.

method declaration:

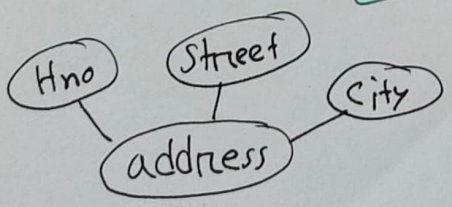
```
method ageOnDate (onDate date)
        returns interval year
        for CustomerType
body { begin
           return onDate - self.dateOfBirth;
       end
```

Constructor Functions:

```
create function Name(f_name varchar(20), l_name
                                varchar(20))
    returns Name
body { begin
           set self.firstname = f_name;
           set self.lastname = l_name;
       end
```

L-20 / 06.11.2024

Slide-11-29 → Coding




```
Create type Address (
        Hno int,
        Street varchar(30),
        City varchar(30))
```

⇒ Create function Address (hno int, street varchar(30), city varchar(30))

```
return Address
begin
    set self.Hno = hno;
    set self.street = street;
    set self.city = city;
end
```

Type inheritance

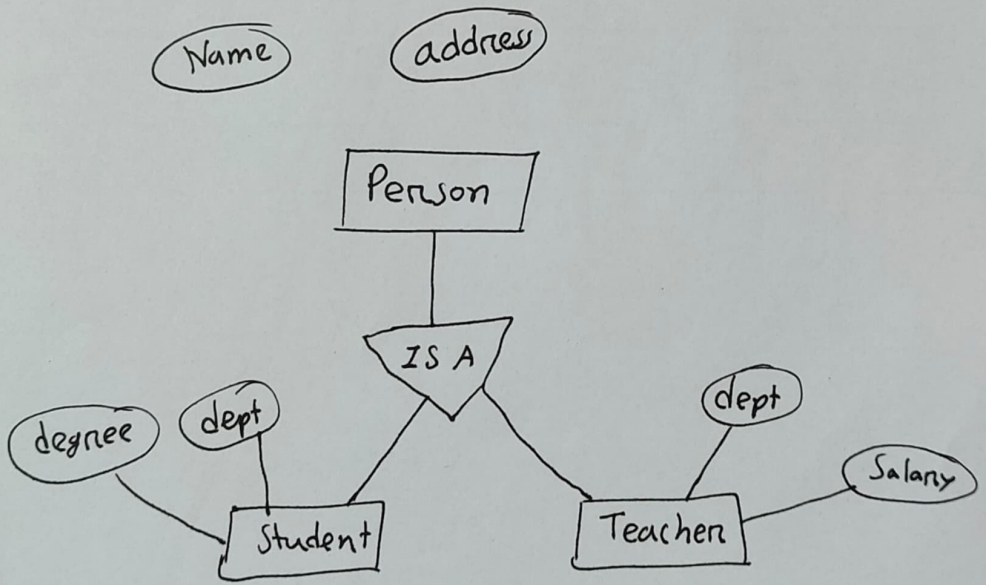


Table Inheritance

- only person :
  - tuples that not contains in lower level.

- Write SQL to find ~~title~~ title for a book containing 'Database' as a keyword.

⇒

```
SELECT title
FROM books
WHERE "Database" IN (unnest (keyword-set))
```

- Unnesting:
  - create all possible tuple list with single value.
  - without array or set.

- Write SQL to find title and keyword for a book containing 'Database' as a keyword.

⇒

```
SELECT B.title, k.keyword
FROM books as B, unnest (B.keyword-set) as k (keyword)
WHERE k.keyword = "Database"
```