15-09-018
CSE501 DBM

purpose of database: 1. Data redundancy & consistancy
- Various file
- Various format
- Various programme

B → A

2 | read (B) → 1000
  | B = B - 150 → 850

4 | write (B) → 850
  | read (A) → 950

6 | A = A + 150 = 1100
  | write (A) → 1100

A → B

1 | read (A) → 1000
  | A = A - 50 → 950

3 | write (A) → 950
  | read (B) → 1000

5 | B = B + 50 → 1050
  | write (B) → 1050

B → A 150
A → B 50

A → 1000 950 1100
B → 1000 850 1050

2000 2150

redundant data and it's redundancy

Student

reg.	Name	Phone
01	AB	017
02	CD	019
03	EF	018

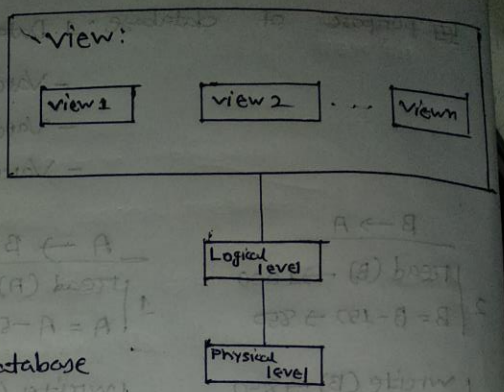
Student reg.

reg	Name	Course
01	AB	CSE501
02	CD	CSE502
03	EF	CSE501

2. Difficult in accessing
3. Data Isolation
4. Integrity problem
5. Automicity problem [অনেকটা বাড়ি করা যায় না, হলে পুরোপুরি হবে, না হলে কিছুই হবে না]
6. Concurrent access anomalies
7. Security problem

View of data: Data abstraction:

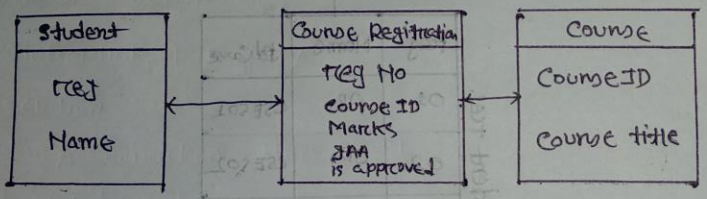
i. Physical level
ii. Logical level
iii. View level



*** Schema & Instance:

Schema: Overall design of the database
(কি কি তথ্য থাকবে হবে)

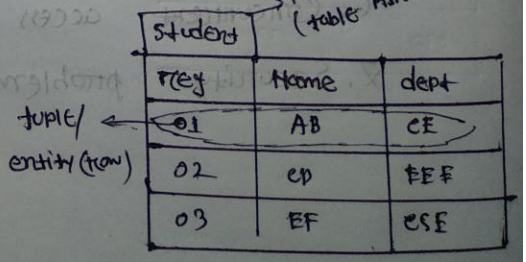
Instance: Collection of information stored in a database at a particular.
(একটি নির্দিষ্ট সময়ের তথ্যগুলোকে Instance বলে)



*** Data Models; 1. Relational Model: Collection of table to represent both data and the relationship among those data. ~~each table~~

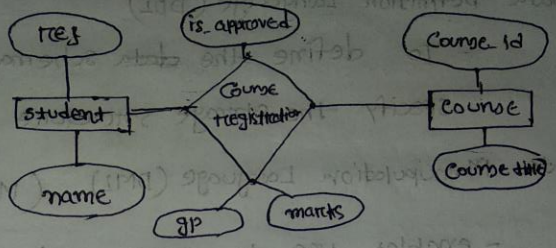
i. each table has multiple columns and each column has a unique name.

ii. Record based model



*** 2: Entity relationship model: – Collection of basic objects
– Widely used in database system

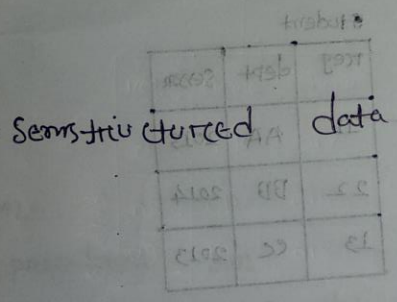
- entity set rectangle
- relationship diamond
- attribute ellipse
- line line

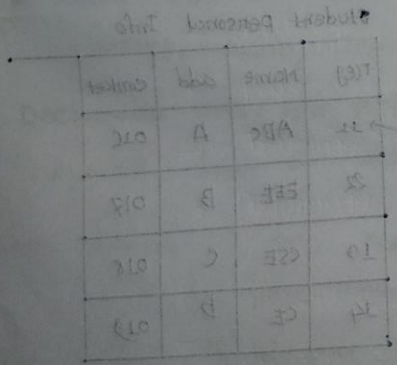


Primary key: কোন Row কে নির্দিষ্টভাবে বোঝার জন্য আমরা যে Column ব্যবহার করি সেটাই primary key. যেমন; কোন reg এর Column এর primary key এর জন্য আমরা কোন নির্দিষ্ট Row খুঁজতে পারি

3. Object based data model: – Object identity
– Object oriented programming
এটি আমরা আগে Chapter 22

4. Semistructured data model: Chapter 23 (এটি আমরা আগে)





22-09-018
DBM 501

Database Language

1. Data Definition Language (DDL)
   - To define the ~~data~~ schema (schem → পরিকল্পনা)
   - specify the storage structure and access method.
2. Data Manupulation Language (DML) (Manipulation → দক্ষতা সহকারে ব্যবহার)
   - enables use to access and modify the data.

```
Creat table student (
 reg  char (20),
 dept char (20),
 Sesson int );
```

reg No	dept.	Session

for DDL

1. Domain constraints
2. Refetential Integrity
3. Assertion
   - A condition db must statisfy be check
4. Authorization
   - i. Read authorization
     - reading but not modification
   - ii. Insert authorization
     - inyert but not modification
   - iii. Update authorization
     - update but not deletion
   - iv. Deletion authorization
     - delete

Student

reg	dept	Sesson
11	AA	2013
22	BB	2014
13	CC	2013

primary key → 11

Student personal Info

reg	Name	add	contact
11	ABC	A	016
22	EEE	B	017
13	CSE	C	018
14	CE	D	019

Foreign key → 11

DML access types:

1. Retrieval (~~fetch~~)
   - Select * from student;
     where reg = 22
2. Insertion
   insert into student ('14', 'AA', 2015)
3. Deletion
   delete from student
   where reg = '13'
4. Modification
   update student set dept = ' '
   Session = 2014 where reg = '12'

DML

i. procedural DML
   * what type of data and how to get these data
     - looping, branching, application, programmer

ii. Declaration DML (Non procedural DML)
   → what types of data without specifying how to get those data

Query process: i. DDL Interpreter
ii. DML capacity
iii. Query Evalution Engine.

Database user

1. Naive user
   - User forms
   - Unsophisticated user
2. Application programmer
   - writes programs using dbs
3. Sophisticated user
   - uses data analysis software
   - data analyst
4. Specized user
   specized db application
   - Computer aided design
   - knowledge base
   - Expert system
   - System that stores data with complex data

Function of DBA

i. Schema definition
ii. Storage structure and access method
iii. Schema and physical organization modification
iv. Granting authorization for data access
v. Routine Maintance
   - Regular Backup
   - Storage checking
   - Monitoring jobs and performence

Introduction to Relational Model

i. Collection of tables with unique name.

ii. The order in which tuples appear in relation in irrele-vant

Student

reg	Name	contact
11	AA	016
22	BB	017
33	CC	018

Marks

reg	Course	gpa
11	CSE 501	4
22	CSE 502	3.75

Table - relation / Entity set

Row - Entity / Tuple

Column - Attribute

Null - Unknown or does not exist corresponding relation

*Database Design and ER Model: Entity Relationship

29-09-018

DBMS

Design Alternatives: - Redundancy

- Incompleteness

i. Entity Set

ii. Relationship set

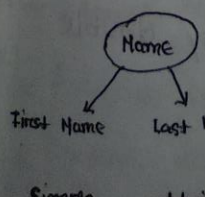
iii. Attributes

Course_reg

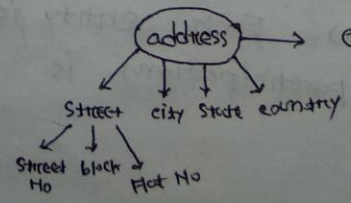
Reg	Course	Title	Credit
11	CSE 333	DB	3
12	CSE 501	DB	3
13	CSE 502	DB LAB	1.5

A ~~data base~~ database is a collection of entity set each of which contains any number of entities and each entity contains a set of attributes.

i. Simple & composit attributes:



→ Component attribute



→ Composit attribut

Simple attributes can not be devided into subparts.

ii. Singled valued and Multiple valued attribute:

- Single value attributes contain single value. For example - Social_Security_Number.
- Multiple value attributes may contains more than one values. For example - a person can have more than one Phone number, email_address etc.

iii. Derived attribute: Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in database. For example average_salary in a department should not be saved directly in the database, insted it can be derived. For another example, age can be derived from date_of_birth.

Mapping Cardinalities:

Cardinality defines the number of entities in one entity set, which can be associated with the number of entities of other set via relationship.

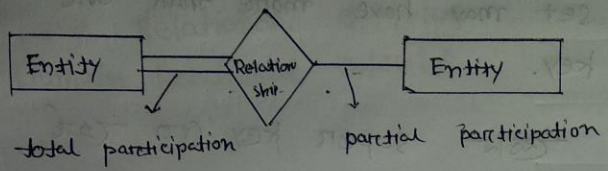
Mapping cardinalities 4 types

i. One to one
ii. One to many
ii. Many to one
iv. Many to many

□ participation constraints

- Total participations. Each entity is involved in the relationship. Total participations is represented by double line.

• partial participations. Not all elements are involved in relationship. partial participations by single lines.



CH #02

keys: key is an attributes or collection of attributes that uniquely identifies an entity among entity set.

For example the roll_number of a student makes him/he identifiable among students.

= uniquely identify tuples

i. Super keys: — Super set of keys
— Set of one or more attributes taken collectively that can uniquely identify a tuple from relation.

pk Instructor

ID	Name	Dept. Name	Phone No.	Salary
10101	MHN	CSE	017	20,000
15151	EH	EEE	018	22,000
16161	SH	PHY	019	25,000

✓ {ID}

✓ {ID, Dept_Name}

✗ {Name}

✓ {Name, dept_Name}

✓ {Name, Phone_No, dept Name}

✓ {Phone number}

✗ {Salary, dept_Name}

- May contains extra attributes
- If K is a super key then any super set of K is super key

2. Candidate key : - Minimal super key
- Super keys for which no proper subset is a super key
- An entity set may have more than one candidate key.

✓ {ID}
× {Name}
× {ID, Dept_Name}
× {Name, Phone_Number, dept_Name}
× {Salary, dept_name}
✓ {Phone number}

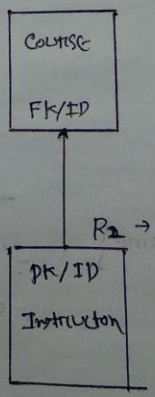
কোন super key থেকে যদি কোন/কতগুলো attribute বাদ দিলে যদি আর super key না থাকে তখন সেটি candidate key.

3. primary key: A primary key is one of the candidate candidate keys chosen by database designer to uniquely identify the entity set/tuple.

4. Foreign key : A relation ~~may~~ R_1 say R_1 may include among it's attribute the primary key of another relation.

Course FK

Course-code	course-title	Session	Semester	Instructor ID
CSE 501	DB	2015	5	15151
CSE 502	DB LAB	2011	5	15151
CSE 401	Graphic	2015	5	16161



CH #02

27-10-018

DBMS

Formal Relational Query Language:

i. Relational Algebra → Add, Sub, Multi etc operations
ii. Tuple Relational calculus
iii. Domain Relational calculus.

Relational Algebra:

i. Duplicate not allowed
ii. Order is irrelevent

Take one or two relations as input and returns a single relation as output.

Fundamental operations:

i. Select → greek letter σ
ii. project → greek letter pi π
iii. Union → greek letter Union $\cup$
iv. Set difference → $-$
v. Cartesian product → $\times$
vi. Rename → greek letter rho ρ
vii. join → $\bowtie$

[Query generate করার যে ভাষা]
[Algebra generate করার যে ভাষা]

i Select operation:

Notation – $\sigma_p(r)$, Here r stands for relation and p is stand for prepositional logic formula which may use connectors like and, or and not. This terms may use relational operator like $-, =, \neq, \geq, <, >, \leq$, and ($\wedge$), or ($\vee$), not

Instructor

id	Name	dept	Salary
10101	MRS	CSE	60,000
20202	MSR	CSE	50,000
15151	HT	EEE	25,000
22222	SS	PHY	30,000

- Find all instructor EEE dept
- Select * From instructor where dept = 'EEE' (Query)
- $\sigma_{dept = "EEE"}(instructor)$ (Algebra)

- Find all instructor of 'CSE' dept whose salary greater than 50,000
- Select * from instructor where dept = "CSE" and salary > 50,000
- $\sigma_{dept = "CSE" \wedge Salary > 50,000}(instructor)$

ii. project operation: It projects colums that satisfy given predicate.

Notation: $\pi_{A_1, A_2 .. A_n}(r)$, where A_1, A_2, A_n are attributes names of relation r.

- List the name of all instructor of 'CSE' dept

- $\pi_{name}(\sigma_{dept = "CSE}(instructor))$

- Select name from instructor where dept = 'ESE'

- List ID, Salary, name of all instructor
- Select ID, Salary, name from instructor

- $\pi_{ID, Salary, name}(instructor)$

iii. Union Operation: It performs binary union between two given relation and defined as $r \cup s = \{t \mid t \in R \text{ or } t \in S\}$

Notation: $r \cup s$

1. r and s must be of same entity [same number of attributes]
2. Domain of ith attribute of s and the ith attribute of r must be same for all.

- Find all instructors who are working either 'EEE' department or in 'PHY' department

- (Select * from instructor where dept = "EEE") Union (Select * from instructor where dept = 'PHY')

- $\sigma_{dept = "EEE"}(instructor) \cup \sigma_{dept = "PHY"}(instructor)$

Teacher

F.K ID	F.K Course ID	F.K Sec ID	F.K Semester	F.K Year
10101	CSE 134	1	Fall	2014
12121	CSE 502	1	Spring	2014
15151	CSE 401	2	Fall	2015
10101	CSE 203	1	Spring	2015

Instructor

PK ID	Name	dept	Salary
10101	MRS	CSE	50,000
12121	MSR	CSE	40,000
15151	HT	EEE	25,000
16161	SS	PHY	30,000

Section

P.K Course ID	P.K Sec ID	P.K Semester	P.K Year	P.K Building
MAT102	1	Spring	2015	EEE
CSE100	2	Spring	2015	CSE
IPE301	1	Summer	2014	CE
CSE134	1	Fall	2014	CSE
CSE200	2	Spring	2015	CSE
CSE200	1	Fall	2014	CSE

* Find the set of all courses tought in Fall 2014 or Spring 2015 or both

- (Select Course_ID from section where Semester = 'Fall' and Year = '2014') Union (Select Course_ID from section where Semester = 'Spring' and Year = 2014)

- $\Pi_{Course\_ID}\left(\sigma_{semester = 'Fall' \wedge year = 2014}(section)\right) \cup \Pi_{Course\_ID}\left(\sigma_{semester = 'spring' \wedge Year = 2015}(section)\right)$

4. Set difference: The result of set difference operation is tuples, which are present in one relation but are not in the second relation.

Notation: $r - s$

Finds all tuples that are present in r but not in s

$\Pi_{author}(Books) - \Pi_{author}(Articles)$

Output: provides name of authors who have written books but not articles.

- Find the set of all courses, ~~the~~ taught in fall 2014 semester but not spring 2015 semester.

- (Select course_ID from section where semester = Fall and year = 2014) MINUS (select course_ID from section where semester = 'spring' and year = 2015)

- $\pi_{courseID}(\sigma_{semester = Fall \wedge year = 2014}(section)) - \pi_{courseID}(\sigma_{semester = spring \wedge year = 2015}(section))$

6. Cartesian product (x)

Combine information of two differenee relation into one

Notation - $r \times s$

- Select * from instructon, teaches where instructon.id = teaches.id.

- $\sigma_{instructon.id = teaches.id}$ (instructon x teaches)

ID	Name	dept	salary
10101	MRS	CSE	50,000
12121	MSR	CSE	40,000
15151	HT	EEE	25,000
10101	MRS	CSE	50,000

ID	course_ID	sec_ID	semeste	year
10101	CSE134	1	Fall	2014
10101	CSE502	1	spring	2014
12121	CSE401	2	Fall	2015
15151	CSE203	1	spring	2015

- Find the name and course_ID of all instructon in phy dept

- Select * ~~from~~ instructon.name, teaches.course_id from instructon, teaches where dept = 'PHY';

- $\pi_{name, course\_Id}(\sigma_{dept = 'PHY' \wedge Instructon.Id = teaches.Id}$ (instructon x teaches))

03-10-018
DBM

* 1000000 SSC info → SSC, reg, board, Roll, GPA

70,000 SSC reg → roll, name, fname, mname

- Select * from applicant-info whecather ssc.reg in (select from ssc. info where board = "Barisal")

Band database

employee (employee_id, name, member, since)
balance (employee_id, balance)
transaction (transaction_id, customer_id, transaction-type, employee-id)
customer (customer_id, name, member_since, account_id
account (account_id, account-type)

- Select name from customer whetre customer_id in (select customer_id from transaction whetre employee_id in (select employee_id from employee whetre name like "Alu chora"));

* join

employee

employee	name	phone
2015001	Tut tut	01
2015323	Tena phuna	02
2015921	peep peep	04

phones

Phone_id	number
01	01912944491
02	017259X212
04	015321C936C

- List the names and Phone of all employee

- Select name, number from employee, phones where employee . phone_id = Phones_ID

- $\Pi_{\text{name, Phone}}\left(\sigma_{\substack{\text{employee. Phone\_Id} \\ = \text{Phones. Phone\_Id}}} (\text{employee} \times \text{phones})\right)$

Naturan join

- Select name, number from employee natural join Phone.

- $\Pi_{\text{name, number}}(\text{employee} \times \text{Phone})$

instructor (Id, Name, dept_name, salary)
section (course_id, sec_id, semester, year, building)
teaches (Id, course_id, sec_id, semester, year)
Student (ID, Name, sec_id, semester, year, cgpa)

- List the courses along with ~~te~~ their ~~buid~~ buildings as well as the instructon for whom the course offend.

- $\Pi_{\text{Id..... Salary, Id..... year, Course\_id.... building}}(\text{instructon} \bowtie \text{teaches} \bowtie \text{Section})$

- ~~Select id.... salary, course_id.... buildings, Id.~~

- Select Id.... Salary, Id..... year, Course_id... buildings from instructon natural join teaches, teaches natural join section.

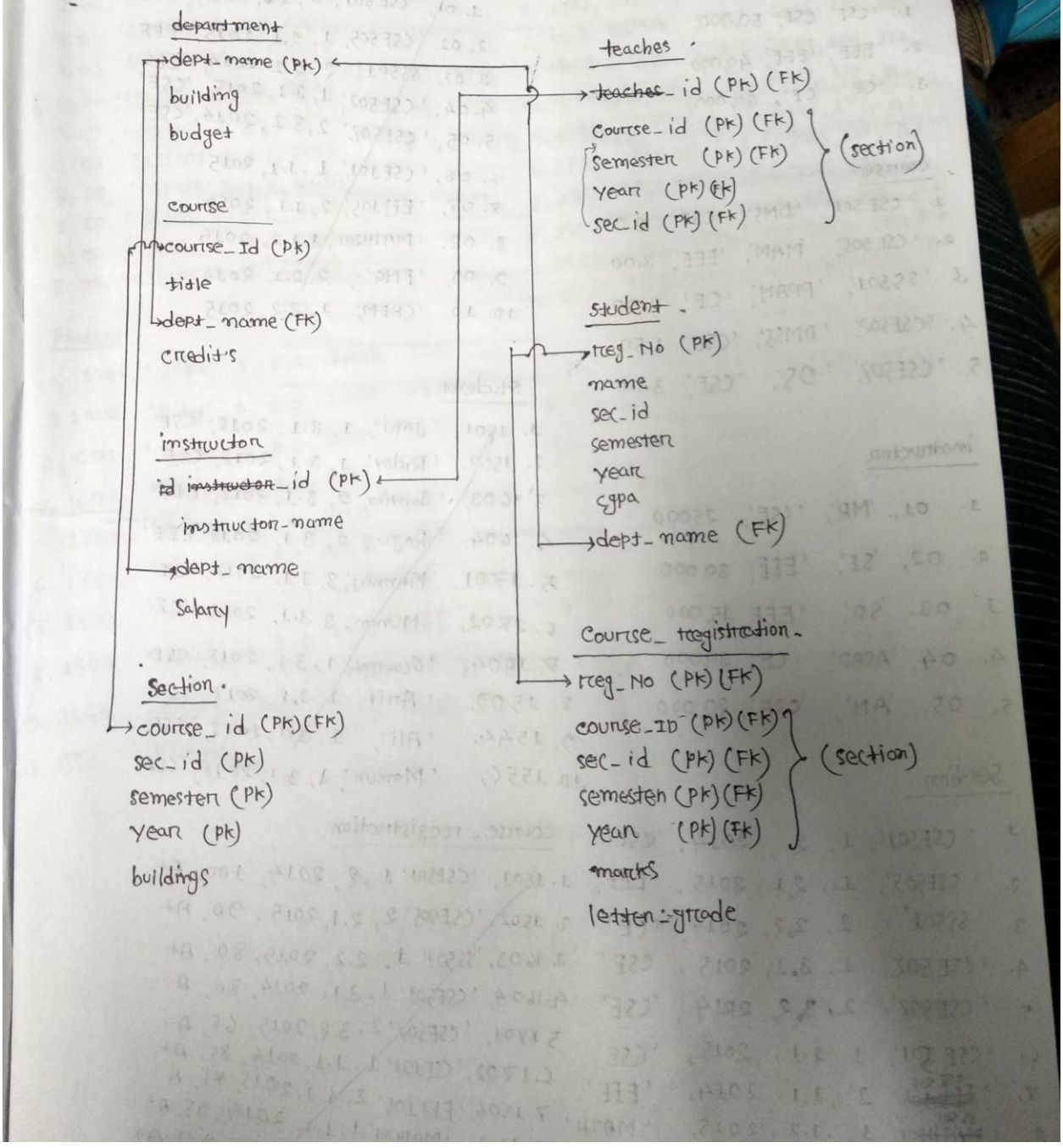
course_offend (course_id, sec_id, semesten, year, buildings name)

course-registration(Id, course_id sec_id, semesten, year, buildings, name, marks, cgpa)

- List the name of the student along with ~~their~~ ~~reesi~~ resisterced courses

- Lab -

Lab_ task


department
dept_name (PK)
building
budget
course
course_Id (PK)
title
dept_name (FK)
credits
instruction
id instructor_id (PK)
instruction_name
dept_name
Salary
Section
course_id (PK)(FK)
sec_id (PK)
semesten (PK)
year (PK)
buildings
teaches
teaches_id (PK) (FK)
course_id (PK) (FK)
Semesten (PK) (FK)
year (PK) (FK)
sec_id (PK) (FK)
(section)
student
reg_No (PK)
name
sec_id
semesten
year
cgpa
dept_name (FK)
Course_ registration
reg_No (PK) (FK)
course_ID (PK) (FK)
sec_id (PK) (FK)
semesten (PK) (FK)
year (PK) (FK)
(section)
marks
letter_grade

* In Values

department:

1. 'CSE', 'CSE', 50,000
2. 'EEE', 'EEE', 40,000
3. 'CE', 'CE', 30,000

course:

1. 'CSE 501', 'DMS', 'CSE', 3.00
2. 'CSE 505', 'MAM', 'EEE', 3.00
3. 'SS501', 'PPAM', 'CE', 2.00
4. 'CSE502', 'DMSS', 'CSE', 1.50
5. 'CSE507', 'OS', 'CSE', 3.00

instructon

1. 01, 'MR', 'CSE', 25000
2. 02, 'SI', 'EEE', 30,000
3. 03. 'SO', 'EEE, 15,000
4. 04, 'ASAD', 'CE', 30,000
5. 05, 'AN', 'CSE', 30,000

section

1. 'CSE501', 1, 2, 2014, 'CSE'
2. 'CSE505', 1, 2.1, 2015, 'EEE'
3. 'SS501', 2, 2.2, 2014, 'CE'
4. 'CSE502', 1, 3.1, 2015, 'CSE'
5. 'CSE507', 2, 3.2, 2014, 'CSE'
6. 'CSE101', 1, 1.1, 2015, 'CSE'
7. ~~'EEE105'~~ CSE 505 'EEE', 2, 1.1, 2014, 'EFE'
8. ~~'MATH201'~~ SS 501 'MATH', 1, 1.2, 2015, 'ENG'
9. ~~'ENG'~~ CSE 'ENG', 2, 1.1, 2014, 'CHEM'

teaches

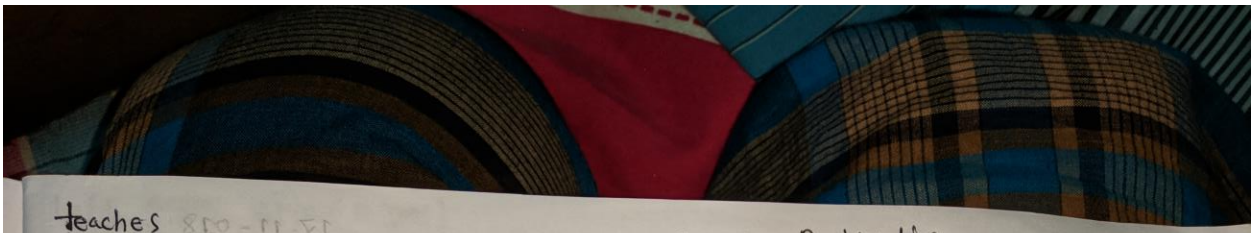
1. 01, 'CSE501', 1, 2.1, 2014, 'CSE'
2. 02, 'CSE505', 1, 2.1, 2015, 'EEE'
3. 03, 'SS501', 2, 2.2 2014, 'CE'
4. 04, 'CSE502', 1, 3.1, 2015, 'CSE'
5. 05, 'CSE507', 2, 3.2, 2014, 'CSE'
6. 06, 'CSE101', 1, 1.1, 2015
7. 07, 'EEE105', 2, 1.1, 2014
8. 08, 'MATH201', 1, 1.2, 2015
9. 09, 'ENG', 2, 2.1, 2014
10. 10, 'CHEM', 1, 2.2, 2015

student

1. 1501, 'Jaki', 1, 3.1, 2018, 'CSE'
2. 1502, 'Ridoy', 1, 3.1, 2018, 'CSE'
3. 1603, 'Imtiaz, 2, 3.1, 2018, 'EEE'
4. 1604, 'Raju', 2, 3.1, 2018, 'EEE'
5. 1701, 'Mannaj', 3, 3.1, 2018, 'CE'
6. 1702, 'Munim', 3, 3.1, 2018, 'CE'
7. 1504, 'Gourab', 1, 3.1, 2018, 'CSE'
8. 1509, 'Amit', 1, 3.1, 2018, 'CE'
9. 1544, 'Ali', 1, 3.1, 2018, 'CSE'
10. 155C, 'Mamun', 1, 3.1, 2018, 'CSE'

course_registration

1. 1501, 'CSE501' 1, 2, 2014, 100, A+
2. 1502, 'CSE505' 2, 2.1, 2015, 90, A+
3. 1603, 'SS501' 1, 2.2, 2015, 80, A+
4. 1604, 'CSE502' 1, 3.1, 2014, 70, A
5. 1701, 'CSE507' 2, 3.2, 2015, 65, A-
6. 1702, 'CSE101' 1, 1.1, 2014, 85, A+
7. 1504, 'EEE105' 2, 1.1, 2015, 75, A
8. 1509, 'MATH201' 1, 1.2, 2014, 95, A+
9. 1544, 'ENG' 2, 2.1, 2015, 90, A+



teaches

1. 01, "CSE501, 1, 2, 2014,
2. 02 'CSE505', 1, 2.1, 2015,
3. 03 'CSE'SS501', 2, 2.2, 2014,
4. 04 'CSE502', 1, 3.1, 2015,
5. 05, 'CSE507', 2, 3.2, 2014,
6. 01, 'CSE501', 1, 1.1, 2015,
7. 02, 'CSE505', 2, 1.1, 2014,
8. 03, 'SS501', 1, 1.2, 2015,
9. 04, 'CSE502', 2, 2.1 2014,
5. 05. 'CSE507', 1, 2.2, 2015,

Student

1. 1501, 'Jaki', 1, 3.1, 2018, 'CSE'
2. 1502, 'Ridoy', 1, 3.1, 2018, 'EEE'
3. 1601, 'Imtiaz', 2, 3.1, 2018, 'CE'
4. 1602, 'Rajo', 2, 3.1, 2018, 'CSE'
5. 1701, 'Afsan', 3, 3.1, 2018, 'EEE'
6. 1702, 'Munim, 3, 3.1, 2018, 'CE'
7. 1503, 'Gourab. 1, 3.1, 2018, 'CSE'
8. 1504, 'Arif', 1, 3.1, 2018, 'EEE'
9. 1544, 'Akbar', 1, 3.1, 2018 'CE'
10. 1556, 'Mamun', 1, 3.1, 2018 'CSE'

Course_Registration

1. 1501, 'CSE501', 1, 2, 2014, 100, A+
2. 1502, 'CSE505', 1, 2.1, 2015, 100, A+
3. 1601 'SS501', 2, 2.2, 2014, 100, A+
4. 1602, 'CSE502', 1, 3.1, 2015, 100, A+
5. 1701, 'CSE507', 2, 3.2, 2014, 100, A+
6. 1702, 'CSE501', 1, 1.1, 2015, 100, A+
7. 1503, 'CSE505', 2, 1.1, 2014, 100, A+
8. 1504 'CSE'SS501', 1, 1.2, 2015, 100, A+
9. 1544, 'CSE502', 2, 2.1, 2014, 100, A+
10. 1556, 'CSE507', 1, 2.2, 2015, 100, A+

17-11-018

Database

Natural join (⋈)

Natural join does not use any comparison operator. It does not concatinate the way a cartesian product does. We can perform a natural join only if there is at least one common attributes that exists between two relations.

~~Student personal~~ info courses

CID	Course	Dept
CS01	Database	CS
CE01	Environment	CE
EE01	Electronics	EE

~~Student academic info~~ HOD

Dept	Head
CS	Samsul
CE	Asad
EE	Iqbal

Courses (⋈) HOD

Dept	CID	Courses	Head
CS	CS01	Database	Samsul
CE	CE01	Environment	Asad
EE	EE01	Electronics	Iqbal

Outer joins:

Natural join is called inner join. An inner join includes only those tuples with matching attributes and the rest are discarded in the resulting relation.

We need to use outer joins to includes all the tuples from the participating relations in the resulting relation. There are three kinds of outer joins. Left outer join, right outer join & full outer join.

Left Outer Join (R ⟕ S)

All the tuples from the left relation, R, are included in the resulting relation. If there are tuples in R without any matching tuple in the Right relation S, then the S-attributes of the resulting relation are made null

Left/Courses

A	B
100	Database
101	Environment
102	Electronics

Right/HoD

A	B
100	Samsul
102	Asad
104	Iqbal

Courses (⟕) HoD

A	B	C	D
100	Database	100	Samsul
101	Environment	–	–
102	Electronics	102	Iqbal

Right Outer Join (R ⟖ S)

All the tuples from the Right relation S, are included in resulting relation. If there are tuples in S without any matching tuples in R, then the R-attributes of resulting relation are made NULL.

Courses (⟖) HoD

A	B	C	D
100	Database	100	Samsul
102	Electronics	102	Asad
–	–	104	Iqbal

Full Outer Join (R ⟗ S)

All the tuples from the both participating relation are included in the resulting relation. If there are no matching tuples for both relation, their respective unmatched attributes are made NULL.

Courses (⟗) HoD

A	B	C	D
100	Database	100	Iq Samsul
101	Environment	–	–
102	Electronics	102	Iqbal Asad
–	–	104	Iqbal

* Outer join

p

a	b	c
e	A	1
f	B	2
g	C	3
h	D	4

q

c	d
2	E
3	F
3	G
5	H

p ⟗ q

a	b	c	d
f	B	2	E
g	C	3	F
g	C	3	G
e	A	1	0
h	D	4	0
0	0	5	H

* Select min (c) from r
* Select count (*) from r
* Select r where c>3

Generalized projection : $\Pi_{name,\ salary*2-200}$

Aggregate function

Aggregate functions are functions that take a collection of values as input and return a single value. SQL offers five built in aggregate functions.

- Average : avg
- Minimum : min
- Maximum : max
- Total : sum
- Count : count

To illustrate the concept of aggregation, we shall use the instructor relation. Suppose we want to find out the sum of salaries of all instructor the relational algebra for this expression for this query is :

$$\mathcal{G}_{sum\ (salary)}\ (instructor)$$

The symbol $\mathcal{G}$ is the letter G in calligraphic font, read it as calligraphic G

instructor

name	dept	salary
ABC	CSE	50,000
DEF	CSE	22,000
GHI	EEE	36,000
JKL	PHY	31,000
MNO	EEE	23,000
PQR	CSE	33,000

dept	total salary
CSE	107000
EEE	59000
PHY	31000

- $\mathcal{G}_{count-distinct(dept)}$

The aggregate function count-distinct ensures that even if an instructor teaches more than one dept, it counted only once in the result.

- Select Salary (sum) from instructor where dept = "CSE"/"EEE"/"PHY"
- Select dept, Salary (sum) as "Total Salary" from instructor
- group by dept order by dept asc/desc.

- Select A_1A_2 sum (A_3) from $r_1, r_2, r_3 \ldots r_n$ where
  p group by A_1, A_2

$$A_1A_2\Pi_{A_1A_2}\left(\mathcal{G}_{sum(A_3)}\left(\sigma_p\left(r_1 \times r_2 \times r_3 \times \cdots \times r_n\right)\right)\right)$$

$$A_1A_2\,\mathcal{G}_{sum(A_3)}\left(\Pi_{A_1A_2}\left(\sigma_p\left(r_1 \times r_2 \times r_3 \times \cdots \times r_n\right)\right)\right)$$

CH # 14

Transaction: Collection of operations that forms a single logical unit of work.

A transaction in a database system must maintain Atomicity, Consistency, Isolation and Durability. Common known as ACID properties. in order to ensure accuracy, completeness and data integrity.

1. Atomicity: Either all operations of the transaction are reflected properly in the database, or none are.

```
T1:  read (A);
     A := A - $50;
     write (A);
     read (B);
     B := B + $50;
     write (B);
```

2. Consistency: Execution of a transaction in isolation (that is, with no other transaction executing concurrently) pe preserves the consistency of data base.

3. Isolation: Even though multiple transaction may execute concurrently, the system guarantees that, for every pair of transaction T_i and T_j, it appears to T_i that either T_j finished execution before T_i started or T_j started execution after T_i finished. Thus, each transaction is unaware of other transaction executing concurrently in the system.

4. Durability: After a transaction completes successfully, the changes it has made to the database persists, even if there are system failures.

These properties are often called the ACID properties; the acronym is derived from the first letter of each of the four properties.

Storage Structure

i. Volatile Storage
- Doesn't survive system crashes
- Extremely fast, direct access to any data
- Example: Memory, Cache

ii. Non-Volatile Storage
- Survive system crashes
- Slower than volatile
- Example: Magnetic, Tape, disk, optical disk, flash drive.

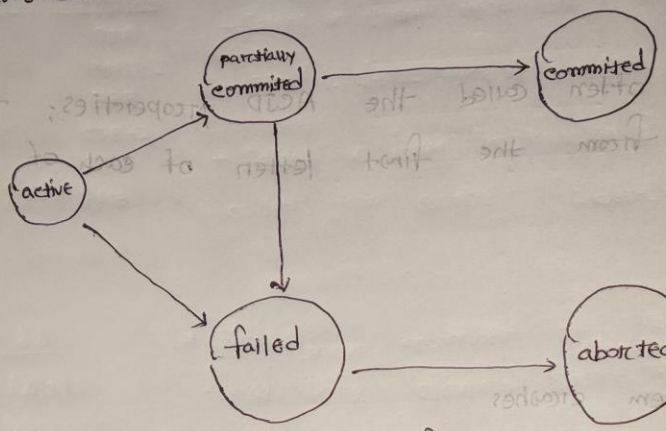
iii. Stable Storage
- Never lost
- Replicate copy in several non volatile storage

Transaction is a unit of program execution that accesses and updates various data items.

Transaction access data using two operations
- read (X)
- write (X)

States of Transactions

A transaction in a database can be one of the following states:



state diagram of transaction

A transaction must be in one of the following states:

- Active: The initial state, the transaction stays in this state while it is executing.
- partially commited: After final statement has been executed.
- Failed: After the discovery that normal execution can no longer proceed.
- Aborted: After the transaction has been rolled back and the database has been stored to its state prior to the start of the transaction.
- Committed: After successful completion.

Advantage of concurrent access

Two advantage of allowing concurrent execution

i. Improved throughput and resource utilization
   - I/O activity
   - CPU processing

ii. Reduced waiting time
   - reduce average response time.

throughput: Number of transaction at a certain time

Serializable

Schedule-1

T_1	T_2
read (A);	
A := 50;	
write (A);	
read (B);	
B := B+50	
write (B)	
	read (A);
	temp := A*0.1;
	A := A-temp
	write (A);
	read (B);
	B := B-temp;
	write (B);

Schedule-2

T_1	T_2
	read (A);
	temp := A*0.1;
	A := A-temp
	write (A);
	read (B);
	B := B-temp;
	write (B)
read (A);	
A := 50;	
write (A);	
read (B);	
B := B+50	
write (B)	

Schedule-3

T_1	T_2
read A;	
A := A-50	
write (A)	
	read (A);
	temp = A*0.1;
	A := A-temp;
	write (A);
read (B);	
B := B+50;	
write (B);	
commit;	
	read (B);
	B := B+temp;
	write (B)
	commit;

* How to determine a transaction is serilazable or most serilazable.

[কোন Resource এর উপর একাধিক transaction যদি conflicting হয় তবে]
Resource এর উপর একাধিক transaction যদি non-conflicting

* Lab Task

Instruction	Course
MAT	STA
ESE	MAT-2
PHY	EEE
	CSE-2
	ETE
	PME

- Select * from instruction natural join course c
- Select * from instruction i inner join course c on (i. dept_name = c. dept_name)
- Select * from instruction i right / left / full outer join course c (i. dept_name = c. dept_name)

aa	bb
(1,2)	(4,2)
(1,2)	(6,3)
(2,3)	(2,5)
(2,5)	(2,7)
(3,2)	(3,5)

* For union number of column same হতে হবে এবং domain same হতে হবে

- (Select * from aa) minus/union (select * from bb)
- Select sum(Y) from aa where x=1;
- Select building, sum (budget) from department group by building order by building asc.

H.W
- inner join
- outer join
- union
- Union all
- minus
- ~~intersect~~ intersect
- aggregate function
- groupe by
- in
- between
- is null
- is not null
- like, _, %
- order by, asc, desc

- Select * from department where budget between 25000 & 35000

Q	T_i	T_j
read(Q)	↓	↓
write(Q)	I	J

1. I = read (Q) J = read (Q) → order doesn't matter
2. I = read (Q) J = write (Q) → order matters
3. I = write (Q) J = read (Q) → order matters
4. I = write (Q) J = write (Q) → order matters

- If two instructions are read operation then their relative order doesn't matter.
- I & J are said to conflict if both words on same data item and at least one of them is write operation.

i. Swap read(B) of T_1 with write (A) of T_2
ii. swap read (B) of T_1 with read (A) of T_2
iii. swap write (B) of T_1 with write (A) of T_2
iv. swap write (B) of T_1 with read (A) of T_2

Schedule - 3

T_1	T_2
read (A);	
write (A);	
	read (A);
	write (A);
read (B);	
write (B);	
	read (B);
	write (B);

- If a Schedule S can be transformed into a Schedule S' by a series of swaps of non-conflicting instructive then S & S' are conflict equivalent.
- If a schedule S in conflict equivalent in serial schedule then it is called conflicted serializable.

Schedule - 5

T_1	T_2
read (A);	
write (A);	
	read (A);
read (B);	
	write (A);
write (B);	
	read (B);
	write (B);

Schedule - 5

T_1	T_2
read (A);	
write (A);	
read (B);	
	read (A);
	write (A);
write (A);	
	read (B);
	read (B);

Schedule - 6

T_1	T_2
read (A);	
write (A);	
read (B);	
write (B);	
	read (A);
	write (A);
	read (B);
	write (B);