IDBMS:

Q What all data is required to design an app for online movie tickets.

- City Names
- Movie Theatre in those respective cities
- Show timings
- Ticket prices
- Movies Available
- Seats Available
- Seat No.
- Movie Category (A/U)
- Movie Length
- Movie Language
- E-mail id
- Screen Name
- Name
- Age
- Phone No.
- Arrangement of seats
- Movie description

(Not all are data → so check it)

Q 3 data that can be used to develop railway application.

→ station names (String)
→ ~~Journey time~~ distance (Float)
→ ~~Stoppages~~ Passenger details (Structure)
→ Ticket Info (Structure)
→ ~~PNR~~ ~~No~~ Train details (Structure)

Data is the atomic unit (cannot be further subdivided).

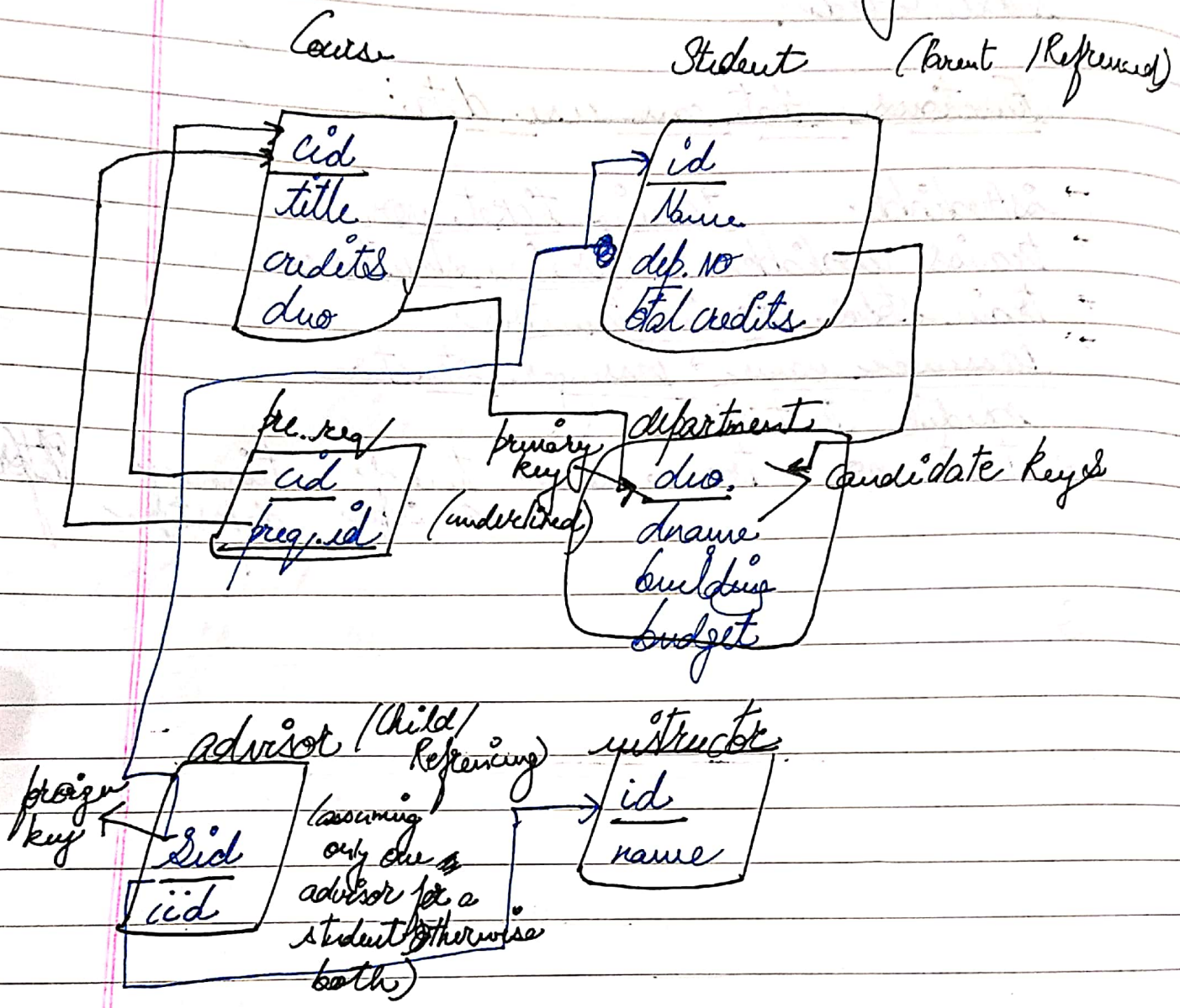
Functions that can use data:

→ isAvailable → using ticket no.
→ trains available → (train structure)
→ train stops → train structure
→ passenger name → passenger structure
schedule (Struct train)
{ train no. : train name time of arrival time of departure }

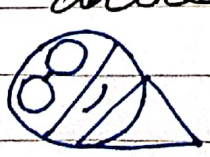
}

21 Aug '19

Relational Model : Key Constraints :



Ex Draw an eq. triangle and draw a circle at one of the sides of Δ is tangent to it. Draw two circles inside the circle ~~of~~ above diameter equi. from each other. Draw a line b/w those circles. Draw a arc in the other semi circle.



but expected

So, we need a model to understand what client

Student

id (Key)	name (Key)	dept.no (Not key)	Total credit (Not key)
92	P	10	160
93	A	20	180
95	K	10	175
40	S	40	95
47	N	30	180

Here tuple 1 is <92, P, 10, 160>

Key: A key is a set of one or more attributes that allows to uniquely identify a tuple in a table.

Note: Here dept.no cannot be a key as 2 dept have same dept.no.

→ Some combination of attributes to uniquely identify,

{id} {name} {id, name} {id, dept}

{id, credit} {name, dept} {name, credit}

{dept, credit, id} {id, name, dept}

{id, name, credit} {id, name, dept, credit}

{name, dept, credit} .....

These are known as superkeys (set of attributes which helps to identify a unique row.

x → are subset of {id, name} → candidate keys. (no proper subset of these)

Superkey (set of attributes)

↓

Candidate key (chosen from superkey)

(Composite) (if more than one)

↓

primary keys (shown underlined in relational model) (chosen by us)

To create relation, we always put primary keys into another table.

Foreign key:

A column which is primary key of another table.

→ Arrow toward primary key

Relational Algebra:

Student

Sid	Name	did	cid	grade
1	Ann	10	C01	A
1	Ann	10	C02	AB
2	Charles	20	C01	A
2	Charles	20	C03	B
2	Charles	20	C04	BC
3	Denis	10	C01	B
4	Frank	20	C02	AB
4	Frank	20	C08	C

→ cid is a foreign key which refers to some course.
→ did is also a foreign key which refers to some department.

Two operations → (Select & Project) Unary
Take relation at a time

Select: Takes input as a relation and output is also a relation

$\sigma_{cond}(R)$
- σ : symbol of selection
- cond : condition
- R : relation name

→ To extract students in CO1.

$\sigma_{cid='co1'}(Student)$
- cid : attribute of R
- 'co1' : value (from domain of attribute)

→ tested for every tuple in the relation

Sid	Name	did	cid	Grade
1	Ann	10	CO1	A
2	Charles	20	CO1	A
3	Dennis	10	CO1	B

← also a relation whose name is not known to us.

→ **Retrieve all students belonging to did = 10**

$\sigma_{did=10}(Student)$

Sid	Name	did	cid	Grade
1	Ann	10	CO1	A
1	Ann	10	CO2	AB
3	Dennis	10	CO1	B

Not duplicate b/c of cid and grade

Select operator suppresses duplicate rows.

→ **did =10 and cid = co1 :**

~~$\sigma_{did=10 \,\&\&\, cid='co1'}$ (Student)~~

Same (commutative)

$\sigma_{did=10}$ ($\sigma_{cid='co1'}$ (Student))

or

$\sigma_{cid='co1'}$ ($\sigma_{did=10}$ (Student))

$$\sigma_{cond1}(\sigma_{cond2}(R)) = \sigma_{cond2}(\sigma_{cond1}(R))$$

$$\sigma_{cond1 \text{ and/or } cond2}(Student)$$

Can use any (and as well as '∧')

∧ / ∨

resting only possible for "and"

$$\sigma_{cond1}(\sigma_{cond2} \dots (\sigma_{cond\,n}(R)) \dots)$$

$$= \sigma_{cond1 \text{ and } cond2 \dots \text{and } cond\,n}(R)$$

→ **Retrieve dept no 10 and 20 :**

$\sigma_{did=10 \text{ or } did=20}$ (Student)

(V) can also use this instead of 'or'

$\sigma_{did=10 \text{ and } did=20}$ (Student) → Wrong b/c the condition checked for each tuple, both should be true for the row which is wrong.

All this is useful in query optimization.

Select * from Student → σ (Student) → condition left blank for all rows and columns.

Project (π):

In select, we take horizontal division of table with all attributes.

In project, we select only some attributes of the table for all rows.

$\pi_{sid, name}$ (Student) → automatically suppresses duplicate rows

Retrieve the name and department of the student who has grade 'A' in any course.

$\pi_{did, name}(\sigma_{grade='A'}(\text{Student}))$

$\sigma_{grade='A'}(\pi_{did, name}(Student))$ ✗

as

did	name

→ has no grade column

Nesting of algebric operators may not be as simple sometimes, so we create a temp table (already was being created, we just name it)

$Temp \leftarrow \sigma_{grade='A'}(Student)$

or

$Temp \leftarrow \sigma_{grade='A'}(Student)$

$Result \leftarrow \pi_{did, name}(Temp)$

or

$Result(\pi_{did, name}(Temp))$

Sign of rename → ρ_S (S: new name) (B_1, B_2, B_3) (new name of attributes) (R) (old name)

If we don't want to rename it, $\rho_S(R)$

(create table, alter table) → The instructions dealing with designing of table are DDL (Data Definition Language)

(insert, delete, update, select) → Instructions dealing with manipulation of data are DML (Data Manipulation Language)

→ Instructions dealing with controlling the data DCL (Data Control Language) (what a user can do, see) (granting / revoking access (everything related to security of data)).

→ Select, from are compulsory part of any select statement. (clauses)

$\pi_{id, name}$ (Student) ≡ Select id, name from Student

→ where (clause) : where there are conditions

select from student where grade = 'A' and cid = 'CO1'
≡ $\sigma_{grade = 'A' \text{ and } cid = 'CO1'}$ (Student)

Sequence of clause execution : from → select → where
(Table) (Attributes, all rows) (Some attributes)

Q. Retrieve sid, name corresponding course id and the grade of that student.

Student (sid, name, did)
Course (cid, cname, credits)
grade (sid, cid, grade)
} Required

Student

sid	name	did
1	a	10
2	a	10
3	b	20

X

Grade

sid	cid	grade
1	C1	A
1	C2	AB
2	C1	A
3	C1	B
3	C2	AB
3	C3	C

= 6 attributes
18 rows

Binary operator

Student X Grade

sid	name	did	sid	cid	grade
1	a	10	1	C1	A
1	a	10	1	C2	AB
1	a	10	2	C1	A
2	a	10	1	C1	A
2	a	10	1	C2	AB
2	a	10	2	C1	A
3	b	20	1	C1	A
3	b	20	1	C2	AB
3	b	20	2	C1	A
1	a	10	3	C1	B
1	a	10	3	C2	AB
1	a	10	3	C3	C
2	a	10	3	C1	B
2	a	10	3	C2	AB
2	a	10	3	C3	C
3	b	20	3	C1	B
3	b	20	3	C2	AB
3	b	20	3	C3	C

$\pi_{sid, name, cid, grade}$ (Student × Grade) → produces all 18 rows with vertical division i.e. only some attributes)

but this would be wrong

Temp1 ← (Student × grade) (18 rows)

Temp2 ← $\sigma_{sid = sid}$ (Temp1) (6 rows)

Temp3 ← $\pi_{sid, name, cid, grade}$ (Temp2) (6 rows, 4 columns)

② select (π) student.sid, name, cid, grade

① from (table) student, grade

③ where (σ) grade.sid = student.sid

→ Select sid, cid corresponding cname, credits and grade for the student.

~~select student.sid, course.cid, cname, credits, grade~~
~~from student, grade, course~~
~~where~~

$$Temp1 \leftarrow (course \times grade)$$

$$Temp2 \leftarrow \sigma_{(course.cid = grade.cid)}(Temp1)$$

$$Temp3 \leftarrow \Pi_{sid, cid, cname, credits, grade}(Temp2)$$

Note: unique keys, support null values

Schema

Q. Retrieve fid, fname with dept_name : 'cse'.

$T_1 \leftarrow$ faculty $\times$ dept

$T_2 \leftarrow \sigma_{\text{faculty.dno} = \text{dept.dno}}(T_1)$

$R \leftarrow \pi_{\text{fid, fname}}\left(\sigma_{\text{name} = \text{'cse'}}(T_2)\right)$

or

$T \leftarrow \sigma_{\text{name} = \text{'cse'}}(\text{dept})$

$T_2 \leftarrow$ faculty $\times$ dept

:

Q. Retrieve (fid, fname, sid, sname, title)

sid, sname, mono, dno, fid fname, dno
fid, sid, title)

Can also be done by product of two at a time

Temp1 ← (project × faculty × student)

Temp2 ← (σ student.sid = project.sid and faculty.fid = project.fid (Temp1)

Temp3 ← π fid, fname, sid, sname, title (Temp2)

Schema: Student (sid, name, mono, dno)
dept (dno, dname, hod)
faculty (fid, fname, dno)
project (fid, sid, title)

Join operator ⋈

Temp1 ← project ⋈ faculty ⋈ student
(project.fid = faculty.fid) (project.sid = student.sid)
C1 C2

Here σ and × are combined.

Natural Join (*)

→ Automatically it takes condition if foreign key and primary key have same name.

R(a, b, (c, d)) * S((c, d), e) → condition is (R.c = S.c and R.d = S.d)
Automatically combines

If we want to natural join on student and dept by dno, then change/rename attribute names.

~~Student~~ I(sid, sname, mono, dno) ← Student(sid, name, mo, dno)

→ Now, we can use natural join.

faculty:

fid	fname	duo
f1	a	d1
f2	b	d1
f3	c	d2
f4	d	d2
f5	e	null

*

dept:

duo	dname	hod
d1	cse	f4
d2	cce	f3
d3	mme	null

Tillnow, we have done inner join.

a	cse
b	cse
c	cce
d	cce

Inner Join

(Right Outer Join)

a	cse
b	cse
c	cce
d	cce
null	mme

Outer Join ⋈

faculty

dname to be shown in result even if no combination

Teacher's Signature

.dio)

a cse
b cse
c cce
d cce
e null

⟕ (left outer join)

Spaces frame to be shown in table even if there is no combination.

We also have (full outer join) also.