More SQL

- Join Expressions
- Views
- Integrity Constraints
- Data types / Schemas
- Auth.
- Triggers
- Recursive Queries
- Aggregation

- date (xxxx-xx-xx)
- time (xx:xx:xx)
- timestamp (date+time)
- interval (period of time)

Join Expressions

– takes two columns & returns another column

Ex: select *
from table
Join table2 on query

~ equal

Select *
from table, table2
where query
{
table.ID = table2.ID

Mega Example:

*Same as Relational Calculus

Relation *course*

course_id	title	dept_name	credits
BIO-301	Genetics	Biology	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3

Relation *prereq*

course_id	prereq_id
BIO-301	BIO-101
CS-190	CS-101
CS-347	CS-101

Not there so null

natural left outer join
course ⟕ prereq

C_ID	title	dep	cred	pre-req
BIO-3	Genetic	Biology	4	BIO-1
CS-19	Game	CS	4	CS-1
CS-3	Robo	CS	3	Null

course **natural full outer join** *prereq*

course_id	title	dept_name	credits	prereq_id
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	*null*
CS-347	*null*	*null*	*null*	CS-101

course **natural right outer join** *prereq*

course ⟖ prereq

course_id	title	dept_name	credits	prereq_id
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-347	*null*	*null*	*null*	CS-101

Whatever side is open is the side that gets preserved.

Syntax

Join types	Join Conditions
inner join	natural
left outer join	on <predicate>
right outer join	using $(A_1, A_1, \ldots, A_n)$
full outer join	

---

create index name on table(key)
→ speed up access to records

User defined types

create type Name as int/numeric/type( ) final

only if no subtypes can be created on type

Domain (can have constraints)

Create domain Name type constraint

Default Values

```
create table student
    (ID varchar (5),
    name varchar (20) not null,
    dept name varchar (20),
    tot cred numeric (3,0) default 0,
    primary key (ID))
```

(1) table type join table2 on query

(2) table condition type join table2

(3) table cond. type join table2 using column
↑ specifies column they have in common to Base of join from.

Jan - Iss 2024

Views – allows you to hide info from users

Ex: create view name as query

Create new table/view based on another

A view of instructors without their salary
```
create view faculty as
    select ID, name, dept_name
    from instructor
```
Find all instructors in the Biology department
```
select name
from faculty
where dept_name = 'Biology'
```
Create a view of department salary totals
```
create view departments_total_salary(dept_name, total_salary) as
    select dept_name, sum (salary)
    from instructor
    group by dept_name
```

Integrity Constraints

- guard against wrong values
  - Not null
  - primary key
  - unique
  - check(P), P predicate

] already covered

Example: ensure that semester is one of fall, winter, spring or summer:
```
create table section (
    course_id varchar (8),
    sec_id varchar (8),
    semester varchar (6),
    year numeric (4,0),
    building varchar (15),
    room_number varchar (7),
    time slot id varchar (4),
    primary key (course_id, sec_id, semester, year),
    check (semester in ('Fall', 'Winter', 'Spring', 'Summer'))
                )
```

Not Covered

- Recur
- Ranking
- Window

Assertions – allow db to express a condition

The sum of all loan amounts for each branch must be less than the sum of all account balances at the branch.

```
create assertion sum-constraint check
    (not exists (select * from branch
                 where (select sum(amount) from loan
                        where loan.branch-name =
                              branch.branch-name)
                 >= (select sum(amount) from account
                     where account.branch-name =
                           branch.branch-name)))
```

Triggers

– executed automatically as side effect

Syntax

Triggering event can be **insert**, **delete** or **update**

Triggers on update can be restricted to specific attributes

- For example, **after update of** *takes* **on** *grade*

Values of attributes before and after an update can be referenced

- **referencing old row as** : for deletes and updates
- **referencing new row as** : for inserts and updates

Triggers can be activated before an event, which can serve as extra constraints. For example, convert blank grades to null.

```
create trigger setnull_trigger before update of takes
referencing new row as nrow
for each row
when (nrow.grade = ' ')
 begin atomic
        set nrow.grade = null;
 end;
```