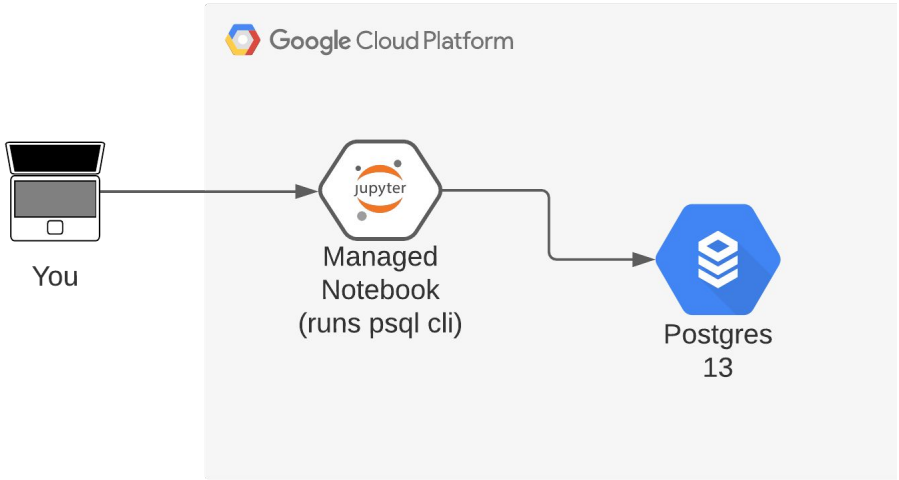
Class 3 Postgres

Elements of Databases

Feb 11, 2022

Postgres environment on Google Cloud



Environment built by following our Postgres and Jupyter setup guides (assigned as homework).

Postgres Overview:

- “The world’s most advanced open source database”
- Implements relational model
- ANSI SQL compliant
- Flexible extension mechanism
- Code base used by research and commercial projects
- Moderately easy to use
- Used for OLTP + (small) OLAP workloads
- Performs on small - medium size data (< TB)
- Performs on small - medium QPS (< 50K)
- Scale reads with read replicas
- Scale writes with application-level sharding

Popular extensions:

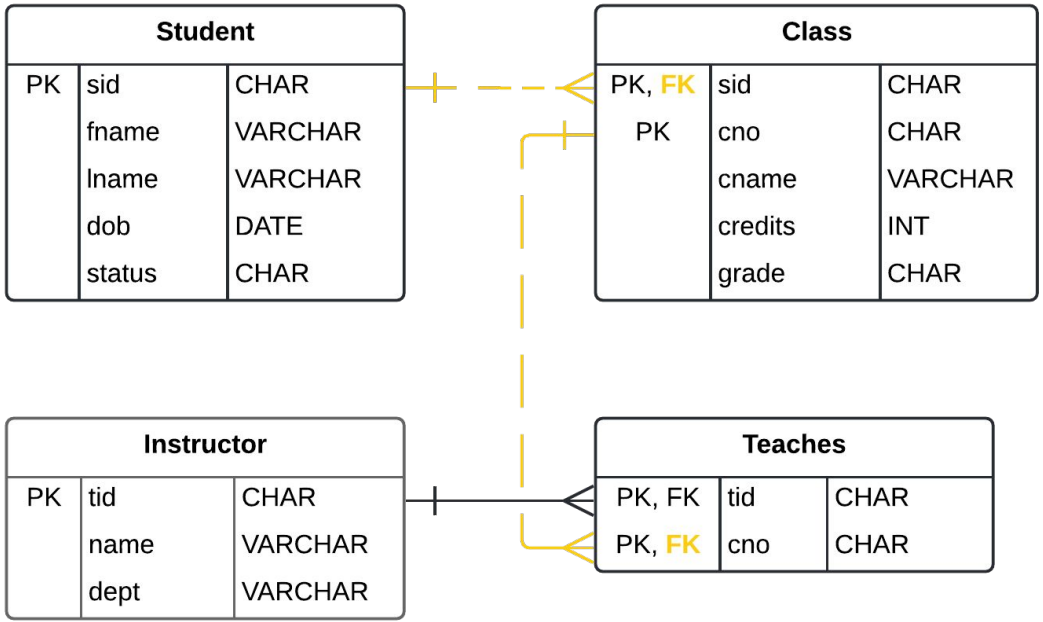
- postgres_fdw
- pgAudit
- HypoPG
- pg_partman

Postgres Code Lab:

- Clone snippets repo
- Open postgres notebook
- Create schema
- Create tables in schema
- Populate tables
- Create Primary Keys
- Create Foreign Keys
- Remodel tables
- Create Primary Keys
- Fix data anomalies
- Create Foreign Keys

Back to our database example…

College data model v1



Two design approaches:

- Bottom-up: Try to create the missing FKs, redesign tables until all FK violations have been resolved.
- Top-down: Identify core business concepts or entities, model them in according to domain requirements while following design guidelines.

Top-down approach example

Domain Requirements:

1. A Student can take zero or more Classes.
2. A Class can have zero or more Students in it.
3. An Instructor can teach zero or more Classes.
4. A Class can be taught by zero or more Instructors.

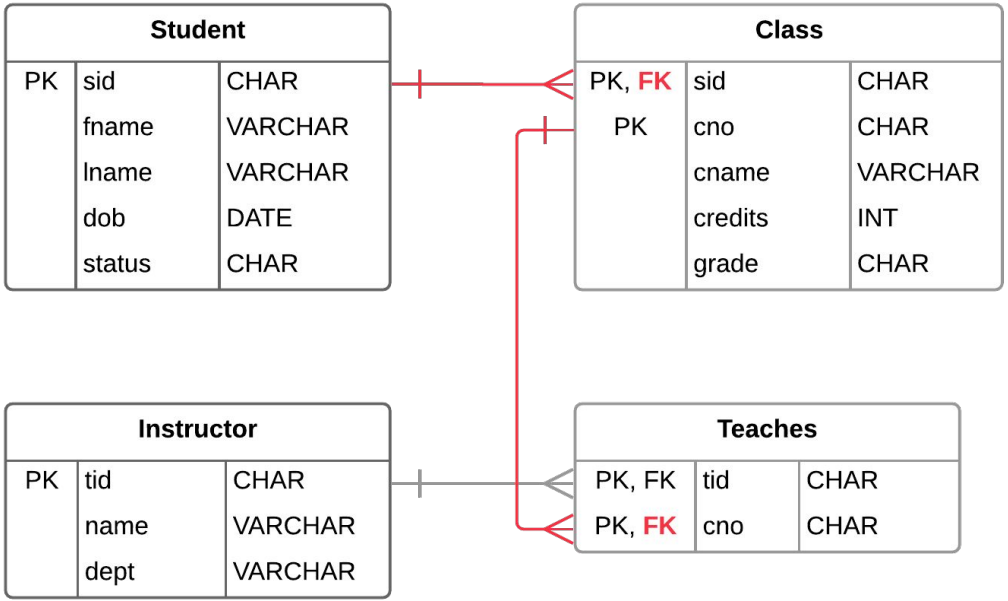
Design guidelines:

A table represents a single entity type or a *m:n* relationship (if junction table).

2. The fields represent the attributes of an entity type or attributes of a *m:n* relationship.
3. Each field is assigned a data type that best fits its domain of values.
4. Each table has a Primary Key (PK) constraint which is made up of one or more fields that uniquely represent each entity in that table.
5. 1:1 and 1:m relationships are represented as a Foreign Key (FK) relationship, in which the child table has a FK constraint on the field(s) that reference its parent's PK fields.

Referential integrity violations

College data model v1

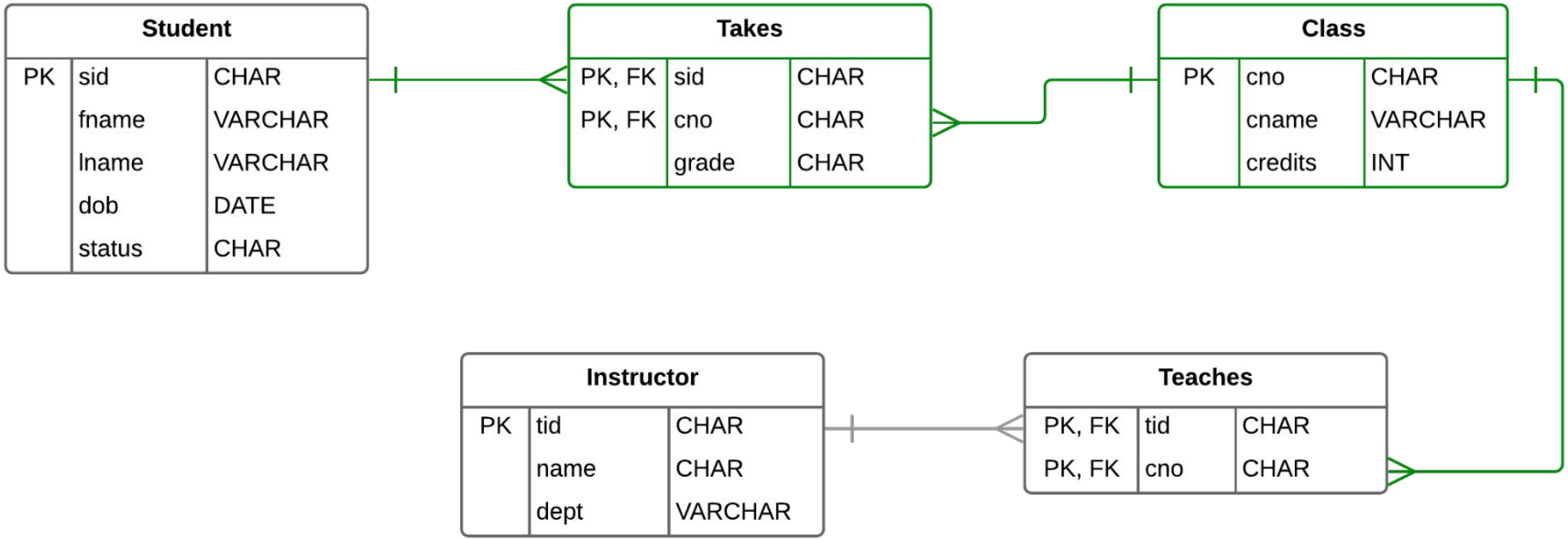


College database anomalies:
- FK on Class.sid
- FK on Teaches.cno

Data anomaly types:
- Insert anomalies
- Update anomalies
- Delete anomalies

Remodeled college database

College data model v2



Implementation Techniques

Use these common data transforms to remodel the tables with standard SQL.

- ```
  CREATE TABLE T2 AS SELECT a, b, c FROM T1;
  ```

- ```
  SELECT a, b, c FROM T1
  UNION [DISTINCT]
  SELECT x AS a, y AS b, z AS c FROM T2;
  ```

- ```
  SELECT a, b, c, 'some string' AS s FROM T1
  UNION ALL
  SELECT d, e, f, 'some string' AS s FROM T2;
  ```
  ```

Join Queries

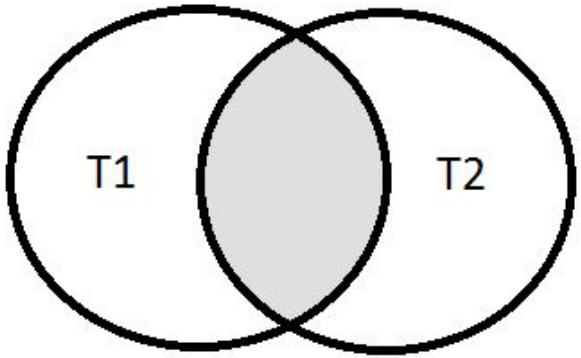
- Queries that use a JOIN operation
- Several flavors of joins
- Pervasive in relational database workloads
- Many optimizations to run efficiently

- Inner joins
- Natural joins
- Outer joins
- Right joins
- Left joins
- Full joins
- Self joins

Inner joins (and Natural joins)

```
SELECT *
FROM T1
[INNER] JOIN T2
ON T1.c1 = T2.c1;

SELECT a.c1, b.c1
FROM T1 a
[INNER] JOIN T1 b
USING c1;
```



Employee

empid	emp_name	*emp_dep*
2	Mike	1
23	Dave	2
3	Sarah	
5	Jim	4
6	Sunil	1
37	Morgan	4

Department

depid	dep_name
1	Sales
2	Product
3	Research
4	Engineering
5	HR

```
SELECT emp_name, dep_name
FROM Employee JOIN Department
ON emp_dep = depid
```

Result Table

emp_name	dep_name
Mike	Sales
Dave	Product
Jim	Engineering
Sunil	Sales
Morgan	Engineering

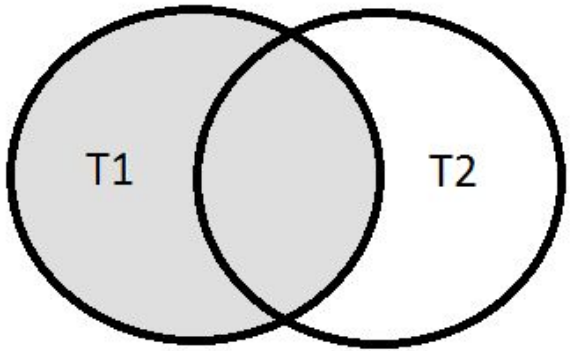
Inner Joins

```
SELECT *
FROM T1
[INNER] JOIN T2 ON T1.c1 = T2.c1
[INNER] JOIN T3 ON T2.c2 = T3.c2;
```

```
SELECT *
FROM T1
[INNER] JOIN T2 ON T1.c1 = T2.c1 AND T1.c2 = T2.c2
[INNER] JOIN T3 ON T2.c2 = T3.c2;
```

Left Outer Joins

```
SELECT *
FROM T1
LEFT [OUTER] JOIN T2
ON T1.c1 = T2.c1;
```



Employee

empid	emp_name	*emp_dep*
2	Mike	1
23	Dave	2
3	Sarah	
5	Jim	4
6	Sunil	1
37	Morgan	4

Department

depid	dep_name
1	Sales
2	Product
3	Research
4	Engineering
5	HR

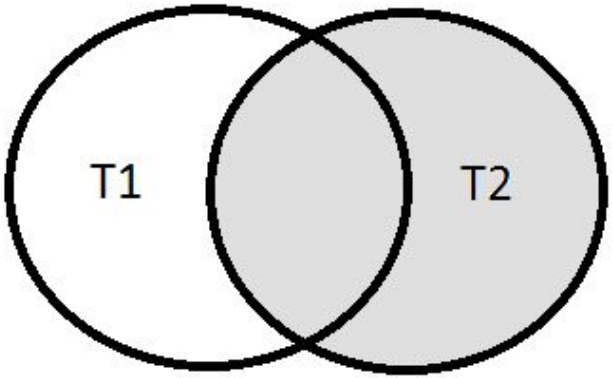
```
SELECT emp_name, dep_name
FROM Employee LEFT JOIN Department ON emp_dep = depid
ORDER BY emp_name
```

Result Table

emp_name	dep_name
Dave	Product
Jim	Engineering
Mike	Sales
Morgan	Engineering
Sarah	
Sunil	Sales

Right Outer Joins

```
SELECT *
FROM T1
RIGHT [OUTER] JOIN T2
ON T1.c1 = T2.c1;
```



Employee

empid	emp_name	*emp_dep*
2	Mike	1
23	Dave	2
3	Sarah	
5	Jim	4
6	Sunil	1
37	Morgan	4

Department

depid	dep_name
1	Sales
2	Product
3	Research
4	Engineering
5	HR

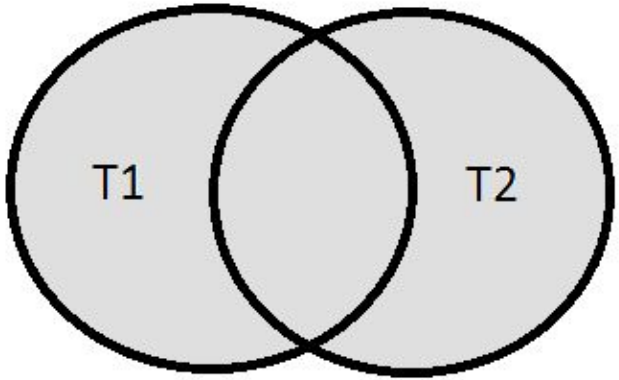
```
SELECT emp_name, dep_name
FROM Employee RIGHT JOIN Department ON emp_dep = depid
ORDER BY dep_name, emp_name
```

Result Table

emp_name	dep_name
Jim	Engineering
Morgan	Engineering
	HR
Dave	Product
	Research
Mike	Sales
Sunil	Sales

Full Outer Joins

```
SELECT *
FROM T1
FULL [OUTER] JOIN T2
ON T1.c1 = T2.c1;
```



Employee

empid	emp_name	*emp_dep*
2	Mike	1
23	Dave	2
3	Sarah	
5	Jim	4
6	Sunil	1
37	Morgan	4

Department

depid	dep_name
1	Sales
2	Product
3	Research
4	Engineering
5	HR

```
SELECT emp_name, dep_name
FROM Employee FULL JOIN Department ON emp_dep = depid
ORDER BY dep_name, emp_name
```

Result Table

emp_name	dep_name
Jim	Engineering
Morgan	Engineering
	HR
Dave	Product
	Research
Mike	Sales
Sunil	Sales
Sarah	

Self Joins

```
SELECT a.c1, b.c1
FROM T1 a
[INNER] JOIN T1 b
ON a.c1 = b.c1;
```

Employee

empid	emp_name	*emp_dep*	dob
2	Mike	1	1990-01-31
23	Dave	2	1983-04-01
3	Sarah		2001-08-02
5	Jim	4	1960-06-13
6	Sunil	1	1979-10-19
37	Morgan	4	1990-01-31

Employee

empid	emp_name	*emp_dep*	dob
2	Mike	1	1990-01-31
23	Dave	2	1983-04-01
3	Sarah		2001-08-02
5	Jim	4	1960-06-13
6	Sunil	1	1979-10-19
37	Morgan	4	1990-01-31

```
SELECT e1.emp_name, e2.emp_name, e2.dob
FROM Employee e1 JOIN Employee e2
ON e1.empid != e2.empid AND e1.dob = e2.dob
```

Result Table

e1.emp_name	e2.emp_name	dob
Mike	Morgan	1990-01-31
Morgan	Mike	1990-01-31

Exercise: SQL Joins

Who are the students who take CS329E with Prof. Mitra?

For each student, return their sid, first and last names, and grade sorted by their sid.

Schema:

Student(<u>sid</u>, fname, lname, dob, status)

Class(<u>cno</u>, cname, credits)

Instructor(<u>tid</u>, name, dept)

Takes(<u>*sid*</u>, <u>*cno*</u>, grade)

Teaches(<u>*tid*</u>, <u>*cno*</u>)

Project 2

http://www.cs.utexas.edu/~scohen/projects/Project2.pdf