UVA | SCHOOL *of* DATA SCIENCE

Database Design

Data Engineering
School of Data Science
University of Virginia

Announcements

- New class material will be posted tonight
- This and lext lectures
  - Database Design
  - Create SQL queries and insert into DBT Data Build Tool
  - Connect DBT with Snowflake to implement database
- Demo, (lab optional) for Docker

UVA | SCHOOL of DATA SCIENCE

Table of Contents

Goals of Effective DB Design

Properly designing a db makes it more useful

A db has CRUD properties: CREATE, READ, UPDATE, DELETE

Considerations include scalability, redundancy, consistency, backups guarding against invalid data, security and permissioning

Transactions should be **atomic**: either all updates happen or not

Database Types

Our work will cover **relational dbs**: contains tables with rows & columns
The tables may be related based on one or more columns

Most dbs are relational. This means they can leverage SQL.

Other types include spreadsheets, hierarchical dbs, XML, Graph dbs, Object dbs, and Document dbs.

The alternative databases can leverage specific data structures (or lack of structure for document db).

Database Types – Relational DBs

Some of the important features:

Data Types – each column uses one data type

Constraints – examples include NON-NULL values, foreign key

Referential Integrity – based on constraints, the db may prevent you from entering invalid data

Joins – related records from multiple tables may be combined

Database Types – Many others..

Node based	Neo4js
Document based	Dynamodb

Relational DB Fundamentals

Tables contain rows (tuples, records) and columns (attributes, fields)
Tables use keys for identifying records

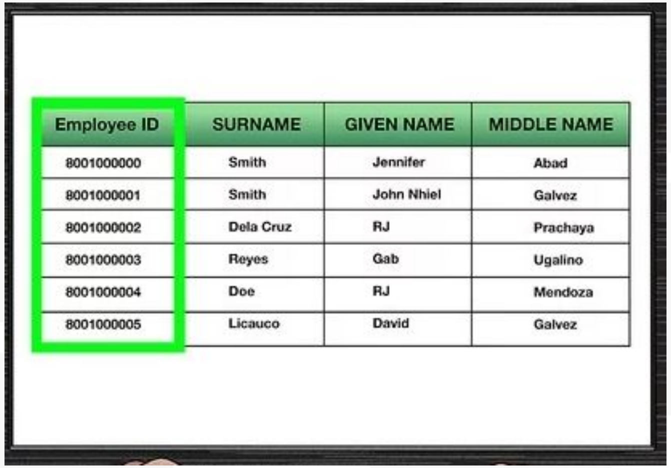
Primary key: one or more fields to uniquely identify records
Foreign key: A key that is used in another table
Indexes: a structure used for efficient search of records
Constraints: restrictions on the data

Next, we provide details for each of these

Relational DB: Primary Key

The primary key is a unique identifier for a record. It can’t be null.

Multiple fields may be used as the primary key (example: ticker, date)



Employee ID	SURNAME	GIVEN NAME	MIDDLE NAME
8001000000	Smith	Jennifer	Abad
8001000001	Smith	John Nhiel	Galvez
8001000002	Dela Cruz	RJ	Prachaya
8001000003	Reyes	Gab	Ugalino
8001000004	Doe	RJ	Mendoza
8001000005	Licauco	David	Galvez

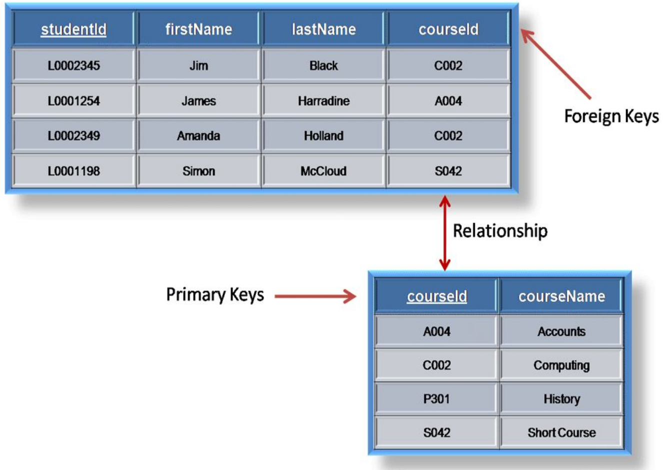
UVA | SCHOOL of DATA SCIENCE

Relational DB: Foreign Key

The bottom table contains course data including the primary key: ***courseId***

The top table includes student data and the courseId.
Since courseId originates in the course table,
it is a foreign key in the top table.

The foreign key helps preserve referential integrity.
Example: If a value isn't in the original table, it cannot be inserted into the table containing the foreign key.



studentId	firstName	lastName	courseId
L0002345	Jim	Black	C002
L0001254	James	Harradine	A004
L0002349	Amanda	Holland	C002
L0001198	Simon	McCloud	S042

courseId	courseName
A004	Accounts
C002	Computing
P301	History
S042	Short Course

Relational DB: Indexes

Indexes allow efficient search (example: index at the back of a book)

Provides a lookup based on one or more fields

Primary key is an index by default

They must be built and maintained. Only build indexes if you’ll use them.

UVA | SCHOOL *of* DATA SCIENCE

Relational DB: Constraints

Basic constraints: not null, invalid data type

Check constraints: field-level constraint, table-level constraint

Primary key constraint: two records in table can’t have same primary key

Uniqueness constraint: can add this requirement to preserve uniqueness

Foreign key constraint: a record’s values in one table must match the values in another table

Understanding User Needs

- Design is a Translation Process
- Bring a List of Questions to understand:
  functionality, data needs, data integrity, security, environment
- Meet the People and Learn Who's Who
  Executive Champion is highest ranking customer driving the project – very important to have one!
- Pick the Customers' Brains
- Get to Understand the Current User's Experience

Understanding User Needs, contd.

Write the Requirements Document:
spell out what you’re planning to build and what it will do

Make Use Cases: this is a script the user can follow to solve a particular, realistic problem they will need to solve.

includes Goals, Summary, Actors, Pre- and post-conditions

Decide Feasibility: take a step back and decide whether the project is feasible table = instructors   table = courses

```
SELECT *
FROM instructors
```

```
name, last-name
efrain, olivares
adam, tashman
…
…
```

Which courses were included in the program, but are no longer active?

```
SELECT COURSE_NAME
FROM PROGRAMS
WHERE active = FALSE
```