# 1 Overview

- **Tuple** - An ordered and named collection of values
  Ex: (`‘Mehta, Dinesh’, ‘CSCI 406’, ‘ALGORITHMS’`)
- **Attributes** - Formally, the attributes are a set
  Ex: (`‘instructor’, ‘course-id’, ‘title’`)
- **Domain** - Each attribute has associated domain from which values come
- **Relation Schema**:
  - A relation schema $R$ is denoted as $R(A_1, A_2, \ldots, A_n)$ where $n$ is the **degree** of the relation and $A_i$ is an **attribute**.
  - Describes how data is stored in a table.
  - Each attribute has a domain that constrains the values that can be associated with it.
- **Relation or Relation State**:
  - A relation is a set of tuples $r(R)$ conforming to the relation schema. In SQL terms, each tuple forms a row.
  - Each element in a tuple maps to an attribute of the schema.
  - As relations are sets, duplicate rows are ignored.
- `NULL` - Some values can me unknown, irrelevant, or missing.
  - NULL values can not be compared with other values.
  - Can only check IS NULL.
- **Constraints** - Limit what values can be present in relation. This can be dependent on the the schema, values in the relation itself, and/or the values in a separate relation.
- **Superkey** - A superkey of $R$ is a subset of $R$'s attributes such that no relation $r(R)$ may contain tuples with exactly the same values for attributes in the superkey.
  - Any superset of a superkey is also a superkey
  - Every relation (in theory) has at least one superkey - the set of all attributes of the schema
- **Key** - A *minimal* superkey of a relation schema. No attribute can be removed without destroying the superkey property.
- **Candidate Keys** - All the keys of a schema are called *candidate keys*.
- **Primary Key** - By convention, we choose one candidate key as the *primary key*.
  - "Smaller" keys are preferred. (e.g. `crn`)
  - In practice in SQL DB:
    * Primary keys impose uniqueness constraint on data.
    * Other candidate keys → impose uniqueness constraints.
- **Referential Integrity/Foreign Key Constraint** - Constraint on the relationship between relations. A *foreign key* is a subset of attributes of a relation schema with the property that its values are either `NULL` or existing in the specified attributes of a referenced table.

# 2 Unary Operations

## 2.1 Selection

- Chooses subsets of tuples from a relation according to some condition (`WHERE`)
- Letting $R$ represent some relation: $\sigma_{\text{condition}}(R)$
- E.g. $\sigma_{\texttt{course\_id = 'CSCI 403'}}(\texttt{mines\_courses})$
- Can use `AND`, `OR`, `NOT`, etc.
- Properties of $\sigma$
  - degree of relation (# of attributes) is the same as the original
  - the number of tuples in the result $\leq$ # tuples in original
  - commutative $\sigma_a(\sigma_b(R)) = \sigma_b(\sigma_a(R))$
  - any sequence of $\sigma$ operations can be replaced with a single $\sigma$ (using `AND`)

## 2.2 Projection

- chooses attributes from set of attributes in a relation
- parallel operation in SQL: `SELECT`
- Notated:

$$\pi_{attr_1, attr_2, \ldots}(R)$$

- Example:

$$\pi_{instructor, course\_id}(mines\_courses)$$

- Properties of $\pi$
  - # of tuples is always $\leq$ # tuples in original
  - **not** commutative: rather

$$\pi_{list_1}(\pi_{list_2}(R)) = \pi_{list_1}(R)$$

  if $list_1 \subset list_2$
  else, ill formed

## 2.3 Renaming - $\rho$

- parallel operation in SQL: `AS`
- Notation:

$$\rho_{S(B_1, B_2, \ldots)}(R)$$

- Example: Let table $X$ have attributes $(a, b, c)$.

$$\rho_{Y(d,e,f)}(X)$$

```
SELECT a AS d,
       b AS e,
       c AS f
  FROM X as Y;
```

## 2.4 Sequences of Operations

### 2.4.1 Representations

1. nesting: example

$$\rho_{(name, courseid)}(\pi_{instructor, courseid}(\sigma_{department='CS'}(mines\_courses)))$$

```
SELECT instructor AS name
       course_id
  FROM mines_courses
 WHERE department = 'CS';
```

2. sequence of named relations

$$R_1 = \sigma_{department='CS'}(mines\_courses)$$
$$R_2 = \pi_{instructor, course\_id}(R_1)$$
$$R_3 = \rho_{(name, course\_id)}(R_2)$$

# 3 Binary Operations

## 3.1 Set Operators

- $A \cup B$ - union
- $A \cap B$ - intersection
- $A - B$ - difference

### 3.1.1 Properties of Set Operators

- Union, intersection are commutative and associative
- Set difference is not commutative or associative

## 3.2 Cartesian Product and Joins

- $A \times B$ - pairs every tuple from $A$ with every tuple from $B$
- If $A$ has $m$ attributes, $B$ has $n$, $A \times B$ has $m+n$ attributes $|A \times B| = |A| \times |B|$.
- Typical Usage:

$$\sigma_{condition}(A \times B) \equiv A \bowtie_{condition} B$$

- Example: $foo \bowtie_{a=b} bar$

## 3.3 Theta Joins

- When:
  - we have $A \bowtie_{cond} B$
  - condition is of general form $cond_1 AND cond_2 AND \ldots$
  - each $cond_n$ $A_i \Theta B_j, A_i \in A, B_i \in B$

  Then we call $A \bowtie_{cond}$ a *theta join*: $A \bowtie_{\Theta} B$
- Further, when $\Theta$ is $=$, then $A \bowtie_{\Theta} B$ is called and *equijoin* which implies that there is some duplicate data column.
- If $A \bowtie_{cond} B$ is an equijoin and condition equates attributes of $A$ with attributes of $B$ of the **same name**, then the join is a *natural join*.
  $A * B$ - books notation
  $A \bowtie B$ - other people's notation
  Example: both `mines_courses` and `mines_courses_meetings` have `crn` field. Then $mines\_courses * mines\_courses\_meetings$. (Note, this automatically projects away duplicate column(s).)

# 4 Completeness

You can demonstrate that: $\sigma, \pi, \cup, -, \times$ is a complete set of operators for relational algebra.

# 5 Odds and Ends

- Division: $\div$ ~inverse of $\times$ (no mirror in SQL)
- Aggregates and grouping: not part of basic algebra