

# **CS 348: Intro to Database Management**

Michael Noukhovitch

Winter 2015, University of Waterloo

Notes written from \*'s lectures.

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	DBMS . . . . .	3
1.1.1	Definitions . . . . .	3
1.1.2	Three-Level Schema . . . . .	3
1.1.3	Interfacing . . . . .	3
1.2	Big Ideas . . . . .	3
1.2.1	Data Independence . . . . .	3
1.2.2	Transaction . . . . .	4
<b>2</b>	<b>Relational Model</b>	<b>4</b>
2.1	Definitions . . . . .	4
2.2	Properties . . . . .	4
2.3	Relations vs SQL Tables . . . . .	4
<b>3</b>	<b>Relation Algebra</b>	<b>5</b>
3.1	Primary Operators . . . . .	5
3.2	Joins . . . . .	5
3.3	Set Operators . . . . .	5
<b>4</b>	<b>SQL</b>	<b>5</b>
4.1	SQL Standard . . . . .	5
4.2	DML . . . . .	6
4.2.1	NULL . . . . .	6
4.2.2	Subquery . . . . .	6

# 1 Introduction

## 1.1 DBMS

### 1.1.1 Definitions

**Database:** a large and persistent collection of data

**DBMS:** a program that manages details for storage and access to a db  
to abstract common functions and create a uniform interface we need:

- **data model:** all data stored uniformly
- **access control:** authorization to modify/view
- **concurrency control:** multiple applications can access at same time
- **database recovery:** nothing is lost
- **database maintenance**

### 1.1.2 Three-Level Schema

**schema:** a description of the data interface to the database

**external schema:** what the app and user see

**conceptual schema:** description of the logical structure of the data

**physical schema:** description of physical aspects (storage algorithms ...)

DBMS allows the data to be stored via the physical schema, reasoned via the conceptual schema, and accessed via the external schema.

### 1.1.3 Interfacing

Interfacing to DBMS, we can interact with it through:

**Data Definition Language:** specifies schemas

- may be different for each schema
- the **data dictionary** (or **catalog**) stores the information

**Data Manipulation Language:** specifies queries and updates (*e.g SQL*)

- navigational (procedural)
- non-navigational (declarative)

## 1.2 Big Ideas

There are three big ideas which have influenced the creation and development of databases

### 1.2.1 Data Independence

**data independence** allows each schema to be independant of the others

- **physical independence:** application immune to changes in storage structure
- **logical independence:** application immune to changes in data organization

### 1.2.2 Transaction

**Transaction:** an application-specified atomic and durable unit of work

**ACID:** transaction properties ensured by the DBMS

- **atomic:** a transaction cannot be split up
- **consistency:** each transaction preserves consistency
- **isolated:** concurrent transaction don't interfere with each other
- **durable:** once completed, changes are permanent

## 2 Relational Model

### 2.1 Definitions

**Relational model:** all information is organized in (flat) relations

- powerful and declarative query language
- semantic integrity constraints (using first order logic)
- data independence

### 2.2 Properties

- based on finite set theory
  - attribute ordering *not strictly necessary*
  - tuples identified by attribute values
  - instance has set semantics *no ordering, no duplicates*
- all attribute values are atomic
- **degree:** number of attributes in schema
- **cardinality:** number of tuples in instance

We can algebraically define databases as a finite set of relation schemas

### 2.3 Relations vs SQL Tables

SQL has extensions on top of the relational model:

1. semantics of instances:
  - relations are **sets** of tuples
  - tables are **multisets** (bags) of tuples
2. unknown values: SQL includes 'null'

## 3 Relation Algebra

### 3.1 Primary Operators

- **Relation Name:**  $R$
- **Selection:**  $\sigma_{condition}(E)$  satisfies some condition
- **Projection:**  $\pi_{attributes}(E)$  only includes these attributes
- **Rename:**  $\rho(R(\bar{F}), E)$ 
  - $\bar{F}$  is a list of  $oldname \mapsto newname$
- **Product:**  $E_1 x E_2$

### 3.2 Joins

- **Conditional Join:**  $E_1 \bowtie_{condition} E_2$
- **Natural Join:**  $E_1 \bowtie E_2$  common attributes

### 3.3 Set Operators

Schemas  $R$  and  $S$  must be **union compatible**: have same number (and type) of fields

- **Union:**  $R \cup S$
- **Difference:**  $R - S$
- **Intersection:**  $R \cap S$

## 4 SQL

### 4.1 SQL Standard

- **Data Manipulation Language:** query and modify tables
- **Data Definition Language:** create tables and enforce access/security

**Example 4.1.** Basic query block

```
select attribute-list
from relation-list
[where condition]
```

## 4.2 DML

### 4.2.1 NULL

A necessary evil that indicates unknown or missing data

- test using `is (not) NULL`
- expressions with NULL e.g. `x + NULL = NULL`
- `where` treats NULL like `False`
- 

### 4.2.2 Subquery

`where` supports predicates as part of its clause

**Example 4.2.** Subquery example

```
select empno, lastname
from employee
where salary >= all
  ( select salary
    from employee )
```

### 4.2.3 Ordering

No ordering can be assumed unless you use `order by`