

CompSci 516
Database Systems

Lecture 2
SQL

Instructor: Sudeepa Roy

Duke CS, Fall 2018

1

Announcements

- If you are enrolled to the class, but have not received the email from Piazza, please send me an email
- HW1 will be released this week
- Project ideas will be posted by next week

Duke CS, Fall 2018

2

Today's topic

- Physical and Logical Data Independence
- Data Model and XML
- More SQL
 - Aggregates (Group-by)!
 - Creating/modifying relations/data
 - Constraints

Acknowledgement:
The following slides have been created adapting the instructor material of the [RG] book provided by the authors Dr. Ramakrishnan and Dr. Gehrke.

Duke CS, Fall 2018

3

Physical and Logical Data Independence

Duke CS, Fall 2018

4

What does a DBMS provide?

Duke CS, Fall 2018

5

Why use a DBMS?

Duke CS, Fall 2019

CompSci 516: Database Systems

6

Why use a DBMS?

Duke CS, Fall 2019 CompSci 516: Database Systems 7

Why use a DBMS?

Duke CS, Fall 2019 CompSci 516: Database Systems 8

When NOT to use a DBMS?

Duke CS, Fall 2019 CompSci 516: Database Systems 9

Levels of Abstractions in a DBMS

- **Physical schema**
 - Storage as files, row vs. column store, indexes
 - will discuss these in later lectures

Duke CS, Fall 2019 CompSci 516: Database Systems 10

Levels of Abstractions in a DBMS

- **Logical/Conceptual schema**
 - describes the stored data in the physical schema
- **Decided by conceptual schema design**
 - e.g. ER Diagram
 - not covered in this course
 - Normalization
 - will be covered

Students(sid: string, name: string, login: string, age: integer, gpa: real)

Duke CS, Fall 2019 CompSci 516: Database Systems 11

Levels of Abstractions in a DBMS

- **External schema**
 - different “views” of the database to different users (later)
- **One physical and logical schema but there can be multiple external schemas**

Duke CS, Fall 2019 CompSci 516: Database Systems 12

Data Independence

- Application programs are insulated from changes in the way the data is structured and stored
- A very important property of a DBMS
- Logical and Physical

Duke CS, Fall 2019

CompSci 516: Database Systems

13

Logical Data Independence

- Users can be shielded from changes in the logical structure of data
- e.g. Students:
`Students(sid: string, name: string, login: string, age: integer, gpa: real)`
- Divide into two relations
`Students_public(sid: string, name: string, login: string)`
`Students_private(sid: string, age: integer, gpa: real)`
- Still a “view” Students can be obtained using the above new relations
 - by “joining” them with sid
- A user who queries this view Students will get the same answer as before

Duke CS, Fall 2019

CompSci 516: Database Systems

14

Physical Data Independence

- The logical/conceptual schema insulates users from changes in physical storage details
 - how the data is stored on disk
 - the file structure
 - the choice of indexes
- The application remains unaltered
 - But the performance may be affected by such changes

Duke CS, Fall 2019

CompSci 516: Database Systems

15

Data Model and XML (an overview)

Duke CS, Fall 2018

16

Data Model

- Applications need to model some real world units
- Entities:
 - Students, Departments, Courses, Faculty, Organization, Employee, ...
- Relationships:
 - Course enrollments by students, Product sales by an organization
- A data model is a collection of high-level data description constructs that hide many low-level storage details

Duke CS, Fall 2019

CompSci 516: Database Systems

17

Data Model

Can Specify:

1. Structure of the data
 - like arrays or structs in a programming language
 - but at a higher level (conceptual model)
2. Operations on the data
 - unlike a programming language, not any operation can be performed
 - allow limited sets of queries and modifications
 - a strength, not a weakness!
3. Constraints on the data
 - what the data can be
 - e.g. a movie has exactly one title

Duke CS, Fall 2019

CompSci 516: Database Systems

18

Important Data Models

- Structured Data
- Semi-structured Data
- Unstructured Data

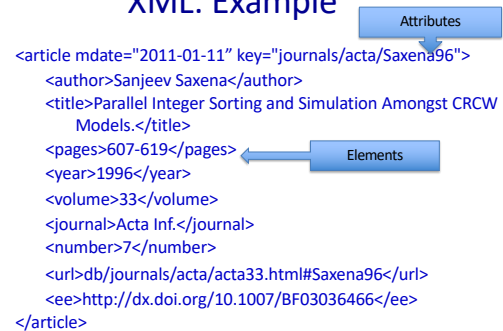
What are these?

Important Data Models

Semi-structured Data and XML

- XML: Extensible Markup Language
- Will not be covered in detail in class, but many datasets available to download are in this form
 - You will download the DBLP dataset in XML format and transform into relational form (in HW1!)
- Data does not have a fixed schema
 - “Attributes” are part of the data
 - The data is “self-describing”
 - Tree-structured

XML: Example



```

<article mdate="2011-01-11" key="journals/acta/Saxena96">
  <author>Sanjeev Saxena</author>
  <title>Parallel Integer Sorting and Simulation Amongst CRCW
    Models.</title>
  <pages>607-619</pages>
  <year>1996</year>
  <volume>33</volume>
  <journal>Acta Inf.</journal>
  <number>7</number>
  <url>db/journals/acta/acta33.html#Saxena96</url>
  <ee>http://dx.doi.org/10.1007/BF03036466</ee>
</article>

```

Attribute vs. Elements

- Elements can be repeated and nested
- Attributes are unique and atomic

XML vs. Relational Databases

+ Serves as a model suitable for integration of databases containing similar data with different schemas

- e.g. try to integrate two student databases: S1(sid, name, gpa) and S2(sid, dept, year)

- Many “nulls” if done in relational model, very easy in XML

- NULL = A keyword to denote missing or unknown values

+ Flexible – easy to change the schema and data

- Makes query processing more difficult

Which one is easier?

- XML (semi-structured) to relational (structured)
- or
- relational (structured) to XML (semi-structured)?

XML to Relational Model

- Problem 1: Repeated attributes

```
<book>
  <author>Ramakrishnan</author>
  <author>Gehrke</author>
  <title>Database Management Systems</title>
  <publisher> McGraw Hill
</book>
```

What is a good relational schema?

Duke CS, Fall 2018

25

XML to Relational Model

- Problem 1: Repeated attributes

```
<book>
  <author>Ramakrishnan</author>
  <author>Gehrke</author>
  <title>Database Management Systems</title>
  <publisher> McGraw Hill</publisher>
</book>
```

Title	Publisher	Author1	Author2

What if the paper has a single author?

Duke CS, Fall 2018

26

XML to Relational Model

- Problem 1: Repeated attributes

```
<book>
  <author>Garcia-Molina</author>
  <author>Ullman</author>
  <author>Widom</author>
  <title>Database Systems – The Complete Book</title>
  <publisher>Prentice Hall</publisher>
</book>
```

Does not work

Title	Publisher	Author1	Author2

Duke CS, Fall 2018

27

XML to Relational Model

Duke CS, Fall 2018

28

XML to Relational Model

Duke CS, Fall 2018

29

Summary: Data Models

- Relational data model is the most standard for database managements
 - and is the main focus of this course
- Semi-structured model/XML is also used in practice – you will use them in hw assignments
- Unstructured data (text/photo/video) is unavoidable, but won't be covered in this class

Duke CS, Fall 2018

30

Back to SQL!

Duke CS, Fall 2018

31

Expressions and Strings

```
SELECT S.age, age1=S.age-5, 2*S.age AS age2
FROM Sailors S
WHERE S.sname LIKE 'B_%B'
```

- Illustrates use of arithmetic expressions and string pattern matching
- Find triples (of ages of sailors and two fields defined by expressions) for sailors
 - whose names begin and end with B and contain at least three characters
- LIKE is used for string matching. `_' stands for any one character and `%' stands for 0 or more arbitrary characters
 - You will need these often

Duke CS, Fall 2019

32

Find sid's of sailors who've reserved a red or a green boat

```
Sailors(sid, sname, rating, age)
Reserves(sid, bid, day)
Boats(bid, bname, color)
```

- Assume a Boats relation
- UNION: Can be used to compute the union of any two *union-compatible* sets of tuples
 - can themselves be the result of SQL queries
- If we replace OR by AND in the first version, what do we get?
- Also available: EXCEPT (What do we get if we replace UNION by EXCEPT?)

Duke CS, Fall 2016

CompSci 516: Data Intensive Computing Systems

Find sid's of sailors who've reserved a red and a green boat

```
Sailors(sid, sname, rating, age)
Reserves(sid, bid, day)
Boats(bid, bname, color)
```

Duke CS, Fall 2016

CompSci 516: Data Intensive Computing Systems

Find sid's of sailors who've reserved a red and a green boat

```
Sailors(sid, sname, rating, age)
Reserves(sid, bid, day)
Boats(bid, bname, color)
```

- INTERSECT: Can be used to compute the intersection of any two *union-compatible* sets of tuples.
 - Included in the SQL/92 standard, but some systems don't support it

Duke CS, Fall 2016

CompSci 516: Data Intensive Computing Systems

Nested Queries

Find names of sailors who've reserved boat #103:

```
SELECT S.sname
FROM Sailors S
WHERE S.sid IN (SELECT R.sid
                FROM Reserves R
                WHERE R.bid=103)
```

```
Sailors(sid, sname, rating, age)
Reserves(sid, bid, day)
Boats(bid, bname, color)
```

- A very powerful feature of SQL:
 - a WHERE/FROM/HAVING clause can itself contain an SQL query
- To find sailors who've not reserved #103, use NOT IN.
- To understand semantics of nested queries, think of a **nested loops evaluation**
 - For each Sailors tuple, check the qualification by computing the subquery

Duke CS, Fall 2018

CompSci 516: Database Systems

36

Nested Queries with Correlation

Find names of sailors who've reserved boat #103:

```
SELECT S.sname
FROM Sailors S
WHERE EXISTS (SELECT *
              FROM Reserves R
              WHERE R.bid=103 AND S.sid=R.sid)
```

- EXISTS is another set comparison operator, like IN
- Illustrates why, in general, subquery must be re-computed for each Sailors tuple

Duke CS, Fall 2018 CompSci 516: Database Systems 37

Nested Queries with Correlation

Find names of sailors who've reserved boat #103 at most once:

```
SELECT S.sname
FROM Sailors S
WHERE UNIQUE (SELECT R.bid
             FROM Reserves R
             WHERE R.bid=103 AND S.sid=R.sid)
```

- If UNIQUE is used, and * is replaced by R.bid, finds sailors with at most one reservation for boat #103
 - UNIQUE checks for duplicate tuples

Duke CS, Fall 2018 CompSci 516: Database Systems 38

More on Set-Comparison Operators

- We've already seen IN, EXISTS and UNIQUE
- Can also use NOT IN, NOT EXISTS and NOT UNIQUE.
- Also available: *op ANY, op ALL, op IN*
 - where *op* : >, <, =, <=, >=
- Find sailors whose rating is greater than that of some sailor called Horatio
 - similarly ALL

```
SELECT *
FROM Sailors S
WHERE S.rating > ANY (SELECT S2.rating
                    FROM Sailors S2
                    WHERE S2.sname='Horatio')
```

Duke CS, Fall 2019 39

Recall: Aggregate Operators

Check yourself: What do these queries compute?

```
COUNT (*)
COUNT ([DISTINCT] A)
SUM ([DISTINCT] A)
AVG ([DISTINCT] A)
MAX (A)
MIN (A)
```

single column

```
SELECT COUNT (*)
FROM Sailors S
```

```
SELECT AVG (S.age)
FROM Sailors S
WHERE S.rating=10
```

```
SELECT S.sname
FROM Sailors S
WHERE S.rating=(SELECT MAX(S2.rating)
               FROM Sailors S2)
```

```
SELECT COUNT (DISTINCT S.rating)
FROM Sailors S
WHERE S.sname='Bob'
```

```
SELECT AVG (DISTINCT S.age)
FROM Sailors S
WHERE S.rating=10
```

Duke CS, Fall 2016 CompSci 516: Data Intensive Computing Systems

Motivation for Grouping

- So far, we've applied aggregate operators to all (qualifying) tuples
 - Sometimes, we want to apply them to each of several groups of tuples
- Consider: Find the age of the youngest sailor for each rating level
 - In general, we don't know how many rating levels exist, and what the rating values for these levels are!
 - Suppose we know that rating values go from 1 to 10; we can write 10 queries that look like this (need to replace i by num):

For $i = 1, 2, \dots, 10$:

```
SELECT MIN (S.age)
FROM Sailors S
WHERE S.rating = i
```

Duke CS, Fall 2018 CompSci 516: Database Systems 41

First go over the examples in the following slides
Then come back to this slide and study yourself

Queries With GROUP BY and HAVING

```
SELECT [DISTINCT] target-list
FROM relation-list
WHERE qualification
GROUP BY grouping-list
HAVING group-qualification
```

- The target-list contains
 - (i) attribute names
 - (ii) terms with aggregate operations (e.g., MIN (S.age))
- The attribute list (i) must be a subset of grouping-list
 - Intuitively, each answer tuple corresponds to a group, and these attributes must have a single value per group
 - Here a group is a set of tuples that have the same value for all attributes in grouping-list

Duke CS, Fall 2018 CompSci 516: Database Systems 42

First go over the examples in the following slides
Then come back to this slide and study yourself

Conceptual Evaluation

- The cross-product of **relation-list** is computed
- Tuples that fail **qualification** are discarded
- 'Unnecessary' fields are deleted
- The remaining tuples are partitioned into groups by the value of attributes in **grouping-list**
- The **group-qualification** is then applied to eliminate some groups
- Expressions in group-qualification must have a **single value per group**
 - In effect, an attribute in **group-qualification** that is not an argument of an aggregate op also appears in **grouping-list**
 - like "...GROUP BY bid, sid HAVING bid = 3"
- One answer tuple is generated per qualifying group

Duke CS, Fall 2018 CompSci 516: Database Systems 43

Find age of the youngest sailor with age >= 18, for each rating with at least 2 such sailors.

```
SELECT S.rating, MIN (S.age) AS minage
FROM Sailors S
WHERE S.age >= 18
GROUP BY S.rating
HAVING COUNT (*) > 1
```

Sailors instance:

sid	sname	rating	age
22	dustin	7	45.0
29	brutus	1	33.0
31	lubber	8	55.5
32	andy	8	25.5
58	rusty	10	35.0
64	horatio	7	35.0
71	zorba	10	16.0
74	horatio	9	35.0
85	art	3	25.5
95	bob	3	63.5
96	frodo	3	25.5

Answer relation:

rating	minage
3	25.5
7	35.0
8	25.5

Duke CS, Fall 2018 CompSci 516: Database Systems 44

Find age of the youngest sailor with age >= 18, for each rating with at least 2 such sailors.

Step 1: Form the cross product: FROM clause
(some attributes are omitted for simplicity)

rating	age
7	45.0
1	33.0
8	55.5
8	25.5
10	35.0
7	35.0
10	16.0
9	35.0
3	25.5
3	63.5
3	25.5

```
SELECT S.rating, MIN (S.age) AS minage
FROM Sailors S
WHERE S.age >= 18
GROUP BY S.rating
HAVING COUNT (*) > 1
```

Duke CS, Fall 2018 CompSci 516: Database Systems 45

Find age of the youngest sailor with age >= 18, for each rating with at least 2 such sailors.

Step 2: Apply WHERE clause

rating	age
7	45.0
1	33.0
8	55.5
8	25.5
10	35.0
7	35.0
10	16.0
9	35.0
3	25.5
3	63.5
3	25.5

rating	age
7	45.0
1	33.0
8	55.5
8	25.5
10	35.0
7	35.0
10	16.0
9	35.0
3	25.5
3	63.5
3	25.5

```
SELECT S.rating, MIN (S.age) AS minage
FROM Sailors S
WHERE S.age >= 18
GROUP BY S.rating
HAVING COUNT (*) > 1
```

Duke CS, Fall 2018 CompSci 516: Database Systems 46

Find age of the youngest sailor with age >= 18, for each rating with at least 2 such sailors.

Step 3: Apply GROUP BY according to the listed attributes

rating	age
7	45.0
1	33.0
8	55.5
8	25.5
10	35.0
7	35.0
10	16.0
9	35.0
3	25.5
3	63.5
3	25.5

rating	age
7	45.0
1	33.0
8	55.5
8	25.5
10	35.0
7	35.0
10	16.0
9	35.0
3	25.5
3	63.5
3	25.5

rating	age
1	33.0
3	25.5
3	63.5
3	25.5
7	45.0
7	35.0
8	55.5
8	25.5
9	35.0
10	35.0

```
SELECT S.rating, MIN (S.age) AS minage
FROM Sailors S
WHERE S.age >= 18
GROUP BY S.rating
HAVING COUNT (*) > 1
```

Duke CS, Fall 2018 CompSci 516: Database Systems 47

Find age of the youngest sailor with age >= 18, for each rating with at least 2 such sailors.

Step 4: Apply HAVING clause

The group-qualification is applied to eliminate some groups

rating	age
7	45.0
1	33.0
8	55.5
8	25.5
10	35.0
7	35.0
10	16.0
9	35.0
3	25.5
3	63.5
3	25.5

rating	age
7	45.0
1	33.0
8	55.5
8	25.5
10	35.0
7	35.0
10	16.0
9	35.0
3	25.5
3	63.5
3	25.5

rating	age
1	33.0
3	25.5
3	63.5
3	25.5
7	45.0
7	35.0
8	55.5
8	25.5
9	35.0
10	35.0

```
SELECT S.rating, MIN (S.age) AS minage
FROM Sailors S
WHERE S.age >= 18
GROUP BY S.rating
HAVING COUNT (*) > 1
```

Duke CS, Fall 2018 CompSci 516: Database Systems 48

Find age of the youngest sailor with age >= 18, for each rating with at least 2 such sailors.

Step 5: Apply SELECT clause
Apply the aggregate operator
At the end, one tuple per group

```
SELECT S.rating, MIN(S.age) AS minage
FROM Sailors S
WHERE S.age >= 18
GROUP BY S.rating
HAVING COUNT(*) > 1
```

rating	age
7	45.0
1	33.0
8	55.5
8	25.5
10	35.0
7	35.0
10	16.0
9	35.0
3	25.5
3	63.5
3	25.5

rating	age
7	45.0
1	33.0
8	55.5
8	25.5
10	35.0
7	35.0
10	16.0
9	35.0
3	25.5
3	63.5
3	25.5

rating	age
1	33.0
3	25.5
3	63.5
3	25.5
7	45.0
7	35.0
8	55.5
8	25.5
9	35.0
10	35.0

rating	minage
3	25.5
7	35.0
8	25.5

Duke CS, Fall 2018 CompSci 516: Database Systems 49

Nulls and Views in SQL

Duke CS, Fall 2018 CompSci 516: Database Systems 50

Null Values

- Field values in a tuple are sometimes
 - **unknown**, e.g., a rating has not been assigned, or
 - **inapplicable**, e.g., no spouse's name
- SQL provides a special value **null** for such situations.

Duke CS, Fall 2018 CompSci 516: Database Systems 51

Standard Boolean 2-valued logic

- True = 1, False = 0
- Suppose X = 5
 - (X < 100) AND (X >= 1) is **T ∧ T = T**
 - (X > 100) OR (X >= 1) is **F ∨ T = T**
 - (X > 100) AND (X >= 1) is **F ∧ T = F**
 - NOT(X = 5) is **¬T = F**
- Intuitively,
 - T = 1, F = 0
 - For V1, V2 ∈ {1, 0}
 - **V1 ∧ V2 = MIN(V1, V2)**
 - **V1 ∨ V2 = MAX(V1, V2)**
 - **¬(V1) = 1 - V1**

Duke CS, Fall 2018 CompSci 516: Database Systems 52

2-valued logic does not work for nulls

- Suppose rating = null, X = 5
- Is rating > 8 true or false?
- What about **AND, OR** and **NOT** connectives?
 - (rating > 8) AND (X = 5)?
- What if we have such a condition in the **WHERE** clause?

Duke CS, Fall 2018 CompSci 516: Database Systems 53

3-Valued Logic For Null

- TRUE (= 1), FALSE (= 0), UNKNOWN (= 0.5)**
 - unknown is treated as 0.5
- Now you can apply rules from 2-valued logic!
 - For V1, V2 ∈ {1, 0, 0.5}
 - **V1 ∧ V2 = MIN(V1, V2)**
 - **V1 ∨ V2 = MAX(V1, V2)**
 - **¬(V1) = 1 - V1**
- Therefore,
 - NOT UNKNOWN = UNKNOWN
 - UNKNOWN OR TRUE = TRUE
 - UNKNOWN AND TRUE = UNKNOWN
 - UNKNOWN AND FALSE = FALSE
 - UNKNOWN OR FALSE = UNKNOWN

Duke CS, Fall 2018 CompSci 516: Database Systems 54

New issues for Null

- The presence of null complicates many issues. E.g.:
 - Special operators needed to check if value IS/IS NOT NULL
 - Be careful!
 - "WHERE X = NULL" does not work!
 - Need to write "WHERE X IS NULL"
- Meaning of constructs must be defined carefully
 - e.g., WHERE clause eliminates rows that don't evaluate to true
 - So not only FALSE, but UNKNOWNs are eliminated too
 - very important to remember!
- But NULL allows new operators (e.g. outer joins)
- Arithmetic with NULL
 - all of +, -, *, / return null if any argument is null
- Can force "no nulls" while creating a table
 - sname char(20) NOT NULL
 - primary key is always not null

Duke CS, Fall 2018 CompSci 516: Database Systems 55

Aggregates with NULL

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

R1

- What do you get for
 - SELECT count(*) from R1?
 - SELECT count(rating) from R1?

Duke CS, Fall 2018 CompSci 516: Database Systems 56

Aggregates with NULL

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

R1

- What do you get for
 - SELECT count(*) from R1?
 - SELECT count(rating) from R1?
- Ans: 3 for both

Duke CS, Fall 2018 CompSci 516: Database Systems 57

Aggregates with NULL

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

R1

sid	sname	rating	age
22	dustin	7	45
31	lubber	null	55
58	rusty	10	35

R2

- What do you get for
 - SELECT count(*) from R1?
 - SELECT count(rating) from R1?
 - Ans: 3 for both
- What do you get for
 - SELECT count(*) from R2?
 - SELECT count(rating) from R2?

Duke CS, Fall 2018 CompSci 516: Database Systems 58

Aggregates with NULL

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

R1

- What do you get for
 - SELECT count(*) from R1?
 - SELECT count(rating) from R1?
- Ans: 3 for both

sid	sname	rating	age
22	dustin	7	45
31	lubber	null	55
58	rusty	10	35

R2

- What do you get for
 - SELECT count(*) from R2?
 - SELECT count(rating) from R2?
- Ans: First 3, then 2

Duke CS, Fall 2018 CompSci 516: Database Systems 59

Aggregates with NULL

- COUNT, SUM, AVG, MIN, MAX (with or without DISTINCT)
 - Discards null values first
 - Then applies the aggregate
 - Except count(*)
- If only applied to null values, the result is null

sid	sname	rating	age
22	dustin	7	45
31	lubber	null	55
58	rusty	10	35

R2

sid	sname	rating	age
22	dustin	null	45
31	lubber	null	55
58	rusty	null	35

R3

- SELECT sum(rating) from R2?
 - Ans: 17
- SELECT sum(rating) from R3?
 - Ans: null

Duke CS, Fall 2018 CompSci 516: Database Systems 60

Creating Relations in SQL

- Creates the "Students" relation
 - the type (**domain**) of each field is specified
 - enforced by the DBMS whenever tuples are added or modified
- As another example, the "Enrolled" table holds information about courses that students take

```
CREATE TABLE Students
(sid CHAR(20),
name CHAR(20),
login CHAR(10),
age INTEGER,
gpa REAL)
```

```
CREATE TABLE Enrolled
(sid CHAR(20),
cid CHAR(20),
grade CHAR(2))
```

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ccs	18	3.2
53650	Smith	smith@math	19	3.8

Students

sid	cid	grade
53831	Carnatic101	C
53831	Reggae203	B
53650	Topology112	A
53666	History105	B

Enrolled

Duke CS, Fall 2018

61

Destroying and Altering Relations

```
DROP TABLE Students
```

- Destroys the relation Students
 - The schema information *and* the tuples are deleted.

```
ALTER TABLE Students
ADD COLUMN firstYear integer
```

- The schema of Students is altered by adding a new field; every tuple in the current instance is extended with a **NULL** value in the new field.

Duke CS, Fall 2018

62

Adding and Deleting Tuples

- Can insert a single tuple using:

```
INSERT INTO Students (sid, name, login, age, gpa)
VALUES (53688, 'Smith', 'smith@ee', 18, 3.2)
```

- Can delete all tuples satisfying some condition (e.g., name = Smith):

```
DELETE
FROM Students S
WHERE S.name = 'Smith'
```

Duke CS, Fall 2018

63

Integrity Constraints (ICs)

- IC**: condition that must be true for **any** instance of the database
 - e.g., **domain constraints**
 - ICs are specified when schema is defined
 - ICs are checked when relations are modified
- A **legal** instance of a relation is one that satisfies all specified ICs
 - DBMS will not allow illegal instances
- If the DBMS checks ICs, stored data is more faithful to real-world meaning
 - Avoids data entry errors, too!

Duke CS, Fall 2018

64

Keys in a Database

- Key / Candidate Key
- Primary Key
- Super Key
- Foreign Key
- Primary key attributes are underlined in a schema
 - Person(pid, address, name)
 - Person2(address, name, age, job)

Duke CS, Fall 2018

65

Primary Key Constraints

- A set of fields is a **key** for a relation if :
 - No two distinct tuples can have same values in all key fields, and
 - This is not true for any subset of the key
- Part 2 false? A **superkey**
- If there are > 1 keys for a relation, one of the keys is chosen (by DBA = DB admin) to be the **primary key**
 - E.g., sid is a key for Students
 - The set {sid, gpa} is a superkey.
- Any possible benefit to refer to a tuple using **primary key** (than any key)?

Duke CS, Fall 2018

66

Primary and Candidate Keys in SQL

- Possibly many **candidate keys**
 - specified using **UNIQUE**
 - one of which is chosen as the primary key.

- “For a given student and course, there is a single grade.”

```
CREATE TABLE Enrolled
(sid CHAR(20)
 cid CHAR(20),
 grade CHAR(2),
 PRIMARY KEY ???)
```

Duke CS, Fall 2018

67

Primary and Candidate Keys in SQL

- Possibly many **candidate keys**
 - specified using **UNIQUE**
 - one of which is chosen as the primary key.

- “For a given student and course, there is a single grade.”

```
CREATE TABLE Enrolled
(sid CHAR(20)
 cid CHAR(20),
 grade CHAR(2),
 PRIMARY KEY (sid, cid))
```

Duke CS, Fall 2018

68

Primary and Candidate Keys in SQL

- Possibly many **candidate keys**
 - specified using **UNIQUE**
 - one of which is chosen as the primary key.

- “For a given student and course, there is a single grade.”

```
CREATE TABLE Enrolled
(sid CHAR(20)
 cid CHAR(20),
 grade CHAR(2),
 PRIMARY KEY (sid, cid))
```

- vs.

- “Students can take only one course, and receive a single grade for that course; further, no two students in a course receive the same grade.”

```
CREATE TABLE Enrolled
(sid CHAR(20)
 cid CHAR(20),
 grade CHAR(2),
 PRIMARY KEY ???,
 UNIQUE ???)
```

Duke CS, Fall 2018

69

Primary and Candidate Keys in SQL

- Possibly many **candidate keys**
 - specified using **UNIQUE**
 - one of which is chosen as the primary key.

- “For a given student and course, there is a single grade.”

```
CREATE TABLE Enrolled
(sid CHAR(20)
 cid CHAR(20),
 grade CHAR(2),
 PRIMARY KEY (sid, cid))
```

- vs.

- “Students can take only one course, and receive a single grade for that course; further, no two students in a course receive the same grade.”

```
CREATE TABLE Enrolled
(sid CHAR(20)
 cid CHAR(20),
 grade CHAR(2),
 PRIMARY KEY sid,
 UNIQUE (cid, grade))
```

Duke CS, Fall 2018

70

Primary and Candidate Keys in SQL

- Possibly many **candidate keys**
 - specified using **UNIQUE**
 - one of which is chosen as the primary key.

- “For a given student and course, there is a single grade.”

```
CREATE TABLE Enrolled
(sid CHAR(20)
 cid CHAR(20),
 grade CHAR(2),
 PRIMARY KEY (sid, cid))
```

- vs.

- “Students can take only one course, and receive a single grade for that course; further, no two students in a course receive the same grade.”

```
CREATE TABLE Enrolled
(sid CHAR(20)
 cid CHAR(20),
 grade CHAR(2),
 PRIMARY KEY sid,
 UNIQUE (cid, grade))
```

- **Used carelessly, an IC can prevent the storage of database instances that arise in practice!**

Duke CS, Fall 2018

71

Foreign Keys, Referential Integrity

- **Foreign key** : Set of fields in one relation that is used to ‘refer’ to a tuple in another relation
 - Must correspond to primary key of the second relation
 - Like a ‘logical pointer’
- E.g. **sid** is a foreign key referring to **Students**:
 - Enrolled(sid: string, cid: string, grade: string)
 - If all foreign key constraints are enforced, **referential integrity** is achieved
 - i.e., no dangling references

Duke CS, Fall 2018

72

Foreign Keys in SQL

- Only students listed in the Students relation should be allowed to enroll for courses

```
CREATE TABLE Enrolled
(sid CHAR(20), cid CHAR(20), grade CHAR(2),
PRIMARY KEY (sid,cid),
FOREIGN KEY (sid) REFERENCES Students )
```

sid	cid	grade
53666	Carnatic101	C
53666	Reggae203	B
53650	Topology112	A
53666	History105	B

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eeecs	18	3.2
53650	Smith	smith@math	19	3.8

Duke CS, Fall 2018

73

Enforcing Referential Integrity

- Consider Students and Enrolled
 - sid in Enrolled is a foreign key that references Students.
- What should be done if an Enrolled tuple with a non-existent student id is inserted?
 - Reject it!
- What should be done if a Students tuple is deleted?
 - Three semantics allowed by SQL
 - Also delete all Enrolled tuples that refer to it (cascade delete)
 - Disallow deletion of a Students tuple that is referred to
 - Set sid in Enrolled tuples that refer to it to a default sid
 - (in addition in SQL): Set sid in Enrolled tuples that refer to it to a special value null, denoting 'unknown' or 'inapplicable'
- Similar if primary key of Students tuple is updated

Duke CS, Fall 2018

74

Referential Integrity in SQL

- SQL/92 and SQL:1999 support all 4 options on deletes and updates.
 - Default is **NO ACTION** (delete/update is rejected)
 - CASCADE** (also delete all tuples that refer to deleted tuple)
 - SET NULL / SET DEFAULT** (sets foreign key value of referencing tuple)

```
CREATE TABLE Enrolled
(sid CHAR(20) DEFAULT '000',
cid CHAR(20),
grade CHAR(2),
PRIMARY KEY (sid,cid),
FOREIGN KEY (sid)
REFERENCES Students
ON DELETE CASCADE
ON UPDATE SET DEFAULT )
```

Duke CS, Fall 2016

CompSci 516: Data Intensive Computing Systems

Where do ICs Come From?

- ICs are based upon the semantics of the real-world enterprise that is being described in the database relations
- Can we infer ICs from an instance?
 - We can check a database instance to see if an IC is violated, but we can **NEVER** infer that an IC is true by looking at an instance.
 - An IC is a statement about **all possible instances!**
 - From example, we know name is not a key, but the assertion that sid is a key is given to us.
- Key and foreign key ICs are the most common; more general ICs supported too

Duke CS, Fall 2018

76

Example Instances

- What does the key (sid, bid, day) in Reserves mean?
- If the key for the Reserves relation contained only the attributes (sid, bid), how would the semantics differ?

sid	sname	rating	age
22	dustin	7	45
31	lubber	8	55
58	rusty	10	35

sid	bid	day
22	101	10/10/96
58	103	11/12/96

Duke CS, Fall 2016

CompSci 516: Data Intensive Computing Systems

Views

- A view is just a relation, but we store a definition, rather than a set of tuples

```
CREATE VIEW YoungActiveStudents (name, grade)
AS SELECT S.name, E.grade
FROM Students S, Enrolled E
WHERE S.sid = E.sid and S.age < 21
```

- Views can be dropped using the **DROP VIEW** command
- Views and Security:** Views can be used to present necessary information (or a summary), while hiding details in underlying relation(s)
 - the above view hides courses "cid" from E
- More on views later in the course

Duke CS, Fall 2018

CompSci 516: Database Systems

78

Can create a new table from a query on other tables too

```
SELECT... INTO... FROM... WHERE
```

```
SELECT S.name, E.grade
INTO YoungActiveStudents
FROM Students S, Enrolled E
WHERE S.sid = E.sid and S.age < 21
```

Duke CS, Fall 2018

CompSci 516: Database Systems

79

“WITH” clause – very useful!

- You will find “WITH” clause very useful!

```
WITH Temp1 AS
  (SELECT .....),
  Temp2 AS
  (SELECT ..... ..)
SELECT X, Y
FROM TEMP1, TEMP2
WHERE....
```

- Can simplify complex nested queries

Duke CS, Fall 2018

CompSci 516: Database Systems

80

Overview: General Constraints

- Useful when more general ICs than keys are involved
- There are also **ASSERTIONS** to specify constraints that span across multiple tables
- There are **TRIGGERS** too : procedure that starts automatically if specified changes occur to the DBMS

```
CREATE TABLE Sailors
(sid INTEGER,
sname CHAR(10),
rating INTEGER,
age REAL,
PRIMARY KEY (sid),
CHECK (rating >= 1
AND rating <= 10))
```

```
CREATE TABLE Reserves
(sname CHAR(10),
bid INTEGER,
day DATE,
PRIMARY KEY (bid,day),
CONSTRAINT noInterlakeRes
CHECK ('Interlake' <>
(SELECT B.bname
FROM Boats B
WHERE B.bid=bid)))
```

Duke CS, Fall 2016

CompSci 516: Data Intensive Computing Systems

Triggers

Only FYI, not covered in detail

- Trigger: procedure that starts automatically if specified changes occur to the DBMS
- Three parts:
 - Event (activates the trigger)
 - Condition (tests whether the triggers should run)
 - Action (what happens if the trigger runs)

```
CREATE TRIGGER youngSailorUpdate
AFTER INSERT ON SAILORS
REFERENCING NEW TABLE NewSailors
FOR EACH STATEMENT
INSERT
INTO YoungSailors(sid, name, age, rating)
SELECT sid, name, age, rating
FROM NewSailors N
WHERE N.age <= 18
```

Duke CS, Fall 2018

CompSci 516: Database Systems

82

Summary: SQL

- SQL has a huge number of constructs and possibilities
 - You need to learn and practice it on your own
 - Given a problem, you should be able to write a SQL query and verify whether a given one is correct
- Pay attention to NULLS
- Can limit answers using “LIMIT” or “TOP” clauses
 - e.g. to output TOP 20 results according to an aggregate
 - also can sort using ASC or DESC keywords

Duke CS, Fall 2018

CompSci 516: Database Systems

83