Administrative Notes – February 28, 2023

- **March 3**: Assignment 4 is due (individual assignment)
  - The question PDF was updated on both Feb 21 and Feb 28 (today) so please make sure you are looking at the right version
  - Tutorials this week will function as open office hours.
    - Friday tutorial moved online for **this week only**
- **March 9**: Midterm
  - During class time. Closed book, closed notes, closed neighbor. More details on Piazza on Thursday.

Administrative Notes – February 28, 2023

- **April 26 @ 8:30AM**: Final exam
  - Location TBD
  - If you have a final exam hardship (3 exams that **start and end** within 24 hours), notify the instructor of the exam in the middle
  - If you have an exam hardship or conflict (two or more exams at the same time), let me know ASAP. UBC rules state you have to notify your instructor 30 days in advance.
  - Final is cummulative

Administrative Notes – February 28, 2023

- Assignment settings mix up
- Assignment 3 was a group assignment but Canvas did not recognize it as such. Unfortunately, we cannot retroactively change assignment settings.
- **Solution**: We created two new assignments that **are** configured for groups and uploaded your submitted work to that assignment.
  - New part 1
  - New part 2

In-Class Exercise (SQL 4)

- Canvas → Modules → In Class Exercises
- You can work on it with other people around you. If you work with others, you must **write their names on your submission to acknowledge the collaboration**.
  - Everyone must submit to Canvas
- Reminder: no late submissions accepted

Clicker Question

I am ready to cover the in-class exercise.

A. Yes
B. No
C. I need two more minutes
D. I need five more minutes

Natural Join

- The SQL NATURAL JOIN is a type of EQUI JOIN and is structured in such a way that, columns with same name of associate tables will appear once only.
- Natural Join : Guidelines
  - The associated tables have one or more pairs of identically named columns.
  - The columns must be the same data type.
  - Don’t use ON clause in a natural join.

```
SELECT *
FROM student s natural join enrolled e
```

- Natural join of tables with no pairs of identically named columns will return the cross product of the two tables.

```
SELECT *
FROM student s natural join class c
```

More fun with joins

- What happens if I execute query:

```
SELECT *
FROM student s, enrolled e
WHERE s.snum = e.snum
```

- To get *all* students, you need an *outer join*
- There are several special joins declared in the *FROM* clause:
  - Inner join – default: only include matches
  - Left outer join – include all tuples from left hand relation
  - Right outer join – include all tuples from right hand relation
  - Full outer join – include all tuples from both relations

Example:

```
SELECT  *
FROM  Student S  NATURAL LEFT OUTER JOIN
Enrolled E
```

More fun with joins examples

R

A	B
1	2
3	3

S

B	C
2	4
4	6

Natural Inner Join

A	B	C
1	2	4

Natural Left outer Join

A	B	C
1	2	4
3	3	Null

Natural Right outer Join

A	B	C
1	2	4
Null	4	6

Natural outer Join

A	B	C
1	2	4
3	3	Null
Null	4	6

Outer join (without the Natural) will use the key word ON for specifying the condition of the join.

Outer join not implemented in MYSQL
Outer join is implemented in Oracle

Clicker Question: Outer

Compute:

```
SELECT R.A, R.B, S.B, S.C, S.D
FROM R FULL OUTER JOIN S
   ON (R.A > S.B AND R.B = S.C)
```

R(A,B)

A	B
1	2
3	4
5	6

S(B,C,D)

B	C	D
2	4	6
4	6	8
4	7	9

Which of the following tuples of R or S is dangling (and therefore needs to be padded in the outer join)?

A. (1,2) of R

B. (3,4) of R

C. (2,4,6) of S

D. All of the above

E. None of the above

Clicker Question: Outer

Start clickerouter.sql

Compute:

```
SELECT R.A, R.B, S.B, S.C, S.D
FROM R FULL OUTER JOIN S
    ON (R.A > S.B AND R.B = S.C)
```

R(A,B)

A	B
1	2
3	4
5	6

S(B,C,D)

B	C	D
2	4	6
4	6	8
4	7	9

Which of the following tuples of R or S is dangling (and therefore needs to be padded in the outer join)?

A. (1,2) of R
B. (3,4) of R
C. (2,4,6) of S
D. All of the above
E. None of the above

A	B	B	C	D
3	4	2	4	6
5	6	4	6	8
1	2	NULL	NULL	NULL
NULL	NULL	4	7	9

Database Manipulation Insertion redux

Can insert a single tuple using:

```
INSERT  INTO  Student VALUES
   (53688, 'Smith', '222 W.15th ave',
   333-4444, MATH)
```

or

```
INSERT  INTO
  Student (sid, name, address, phone,
  major) VALUES  (53688, 'Smith', '222
  W.15th ave', 333-4444, MATH)
```

Add a tuple to student with null address and phone:

```
INSERT  INTO  Student (sid, name,
address, phone, major) VALUES  (33388,
'Chan', null, null, CPSC)
```

Database Manipulation Insertion redux (cont)

- Can add values selected from another table
- Enroll student 51135593 into every class taught by faculty 90873519

```
INSERT  INTO  Enrolled
     SELECT  51135593, name
     FROM Class
     WHERE fid = 90873519
```

The SELECT-FROM-WHERE statement is fully evaluated before any of its results are inserted or deleted.

Database Manipulation Deletion

- Note that only whole tuples are deleted.
- Can delete all tuples satisfying some condition (e.g., name = Smith):

```
DELETE FROM Student
WHERE name = 'Smith'
```

- The WHERE clause can contain nested queries

Database Manipulation Updates

- Increase the age of all students by 2 (should not be more than 100)
- Need to write two updates:

```
UPDATE Student
SET age = 100
WHERE age >= 98

UPDATE Student
SET age = age + 2
WHERE age < 98
```

- Is the order important?

A: Yes

B: No

Integrity Constraints (Review)

- An IC describes conditions that every *legal instance* of a relation must satisfy.
  - Inserts/deletes/updates that violate IC's are disallowed.
  - Can ensure application semantics (e.g., *sid* is a key), or prevent inconsistencies (e.g., *sname* has to be a string, *age* must be < 200)

- *Types of IC's*:
  - domain constraints,
  - primary key constraints,
  - foreign key constraints,
  - general constraints

General Constraints: Check

We can specify constraints over a single table using table constraints, which have the form:

```
CHECK conditional-expression
```

```
CREATE TABLE   Student
     (snum  INTEGER,
      sname  CHAR(32),
      major  CHAR(32),
      standing CHAR(2),
      age  REAL,
      PRIMARY KEY  (snum),
      CHECK  (age >= 10 AND age < 100);
```

Check constraints are checked when tuples are inserted or modified

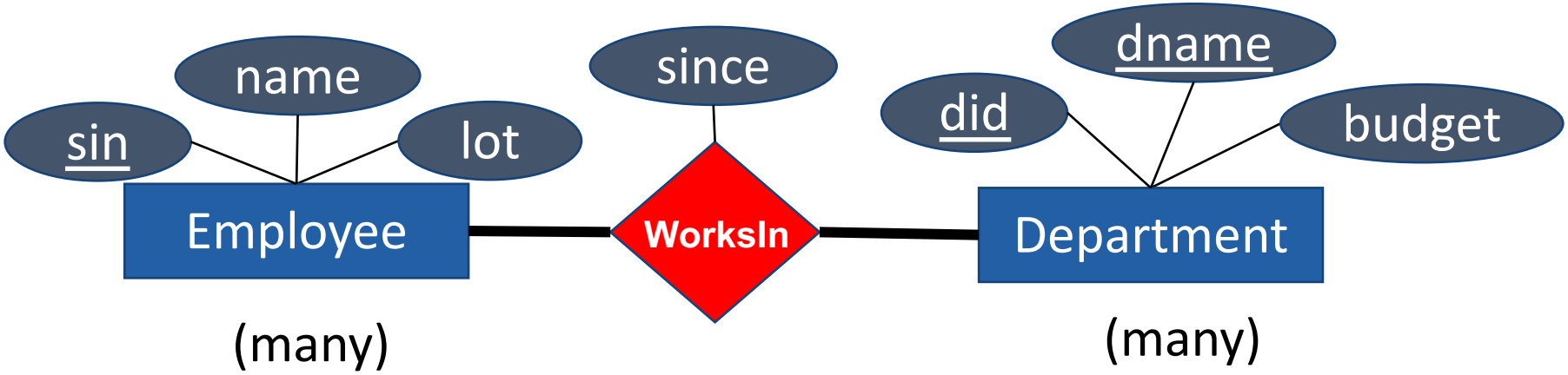
General Constraints: Check

- Constraints can be named
- Can use subqueries to express constraint
- Table constraints are associated with a single table, although the conditional expression in the check clause can refer to other tables

No one can be enrolled in a class, which is held in R15

```
CREATE TABLE  Enrolled
    (snum INTEGER,
     cname   CHAR(32),
     PRIMARY KEY  (snum, cname),
     CONSTRAINT  noR15
        CHECK  (`R15' <>
              (SELECT c.room
               FROM     class c
               WHERE  c.name=cname)));
```

Constraints over Multiple Relations: Remember this one?



- We couldn't express "every employee works in a department and every department has some employee in it"?
- Neither foreign-key nor not-null constraints in WorksIn can do that.
- Assertions to the rescue!

Constraints Over Multiple Relations

- Cannot be defined in one table.
- Are defined as ASSERTIONs which are not associated with any table

- Example: *Every MovieStar needs to star in at least one Movie*

```
CREATE ASSERTION  totalEmployment
CHECK
(NOT EXISTS ((SELECT StarID FROM MovieStar)
              EXCEPT
              (SELECT StarID FROM StarsIn)));
```

Constraints Over Multiple Relations

Example: Write an assertion to enforce every student to be registered in at least one course.

```
CREATE ASSERTION  Checkregistry
CHECK
(NOT EXISTS (
      (SELECT snum FROM student)
      EXCEPT
      (SELECT snum FROM enrolled)));
```

Student(snum,sname,major,standing,age)
Class(name,meets_at,room,fid)
Enrolled(snum,cname)
Faculty(fid,fname,deptid)

Triggers

Useful knowledge
Not tested on exams

- Trigger : a procedure that starts automatically if specified changes occur to the DBMS
- Active Database: a database with triggers
- A trigger has three parts:
  1. Event (activates the trigger)
  2. Condition (tests whether the trigger should run)
  3. Action (procedure executed when trigger runs)

- Database vendors did not wait for trigger standards! So trigger format depends on the DBMS
- NOTE: triggers may cause cascading effects.

Good way to shoot yourself in the foot

That's nice. But how do we code with SQL?

- Direct SQL is rarely used: usually, SQL is embedded in some application code.
- We need some method to reference SQL statements.
- But: there is an *impedance mismatch* problem.
  - Structures in databases <> structures in programming languages
- Many things can be explained with the impedance mismatch.

The Impedance Mismatch Problem

The host language manipulates variables, values, pointers SQL manipulates relations.

There is no construct in the host language for manipulating relations. See https://en.wikipedia.org/wiki/Object-relational_impedance_mismatch

Why not use only one language?

- Forgetting SQL: “we can quickly dispense with this idea” [Ullman & Widom, pg. 363].
- SQL cannot do everything that the host language can do.

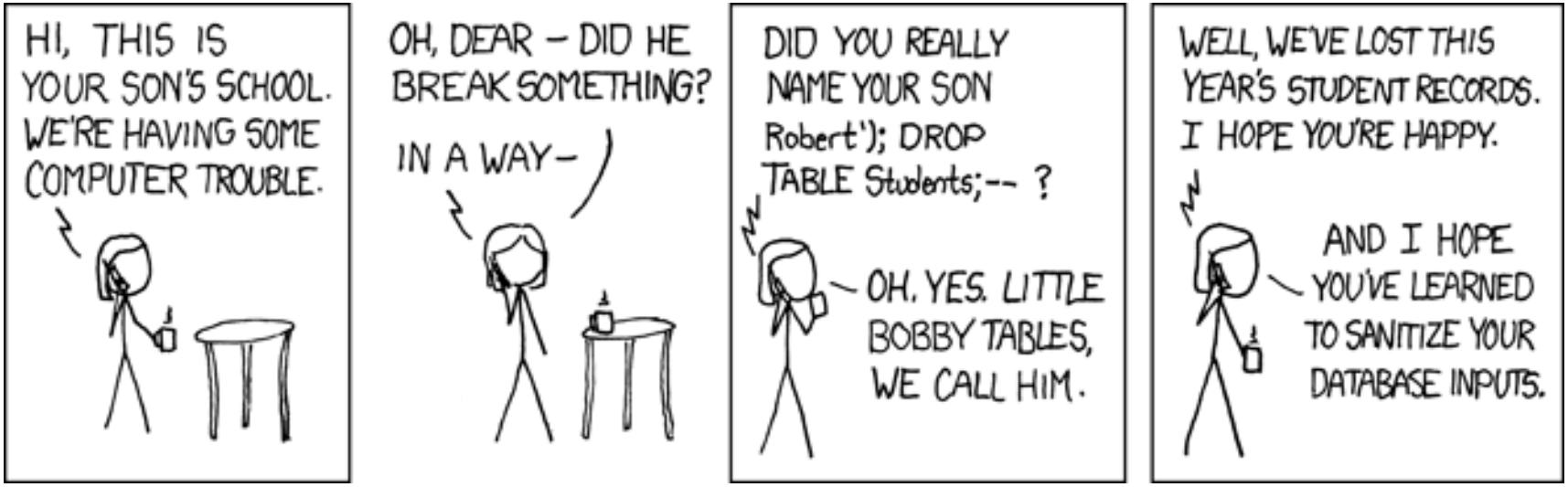
Database APIs

Rather than modify compiler, add library with database calls (API)

- Special standardized interface: procedures/objects
- Passes SQL strings from language, presents result sets in a language-friendly way – solves that impedance mismatch
- Microsoft's *ODBC* is a C/C++ standard on Windows
- Sun's *JDBC* a Java equivalent
- API's are DBMS-neutral
  - a "driver" traps the calls and translates them into DBMS-specific code

And now a brief digression

Have you ever wondered why some websites don't allow special characters?



Summary

- SQL was an important factor in the early acceptance of the relational model; more natural than earlier, procedural query languages.
- Relationally complete; in fact, significantly more expressive power than relational algebra.
- Consists of a data definition, data manipulation and query language.
- Many alternative ways to write a query; optimizer should look for most efficient evaluation plan.
  - In practice, users need to be aware of how queries are optimized and evaluated for best results.

Summary (Cont')

- NULL for unknown field values brings many complications
- SQL allows specification of rich integrity constraints (and triggers)
- Embedded SQL allows execution within a host language; cursor mechanism allows retrieval of one record at a time
- APIs such as ODBC and JDBC introduce a layer of abstraction between application and DBMS

Learning Goals Revisited

- Given the schemas of a relation, create SQL queries using: SELECT, FROM, WHERE, EXISTS, NOT EXISTS, UNIQUE, NOT UNIQUE, ANY, ALL, DISTINCT, GROUP BY and HAVING.
- Show that there are alternative ways of coding SQL queries to yield the same result. Determine whether or not two SQL queries are equivalent.
- Given a SQL query and table schemas and instances, compute the query result.
- Translate a query between SQL and RA.
- Comment on the relative expressive power of SQL and RA.
- Explain the purpose of NULL values and justify their use. Also describe the difficulties added by having nulls.
- Create and modify table schemas and views in SQL.
- Explain the role and advantages of embedding SQL in application programs.
- Write SQL for a small-to-medium sized programming application that requires database access.
- Identify the pros and cons of using general table constraints (e.g., CONSTRAINT, CHECK) and triggers in databases.