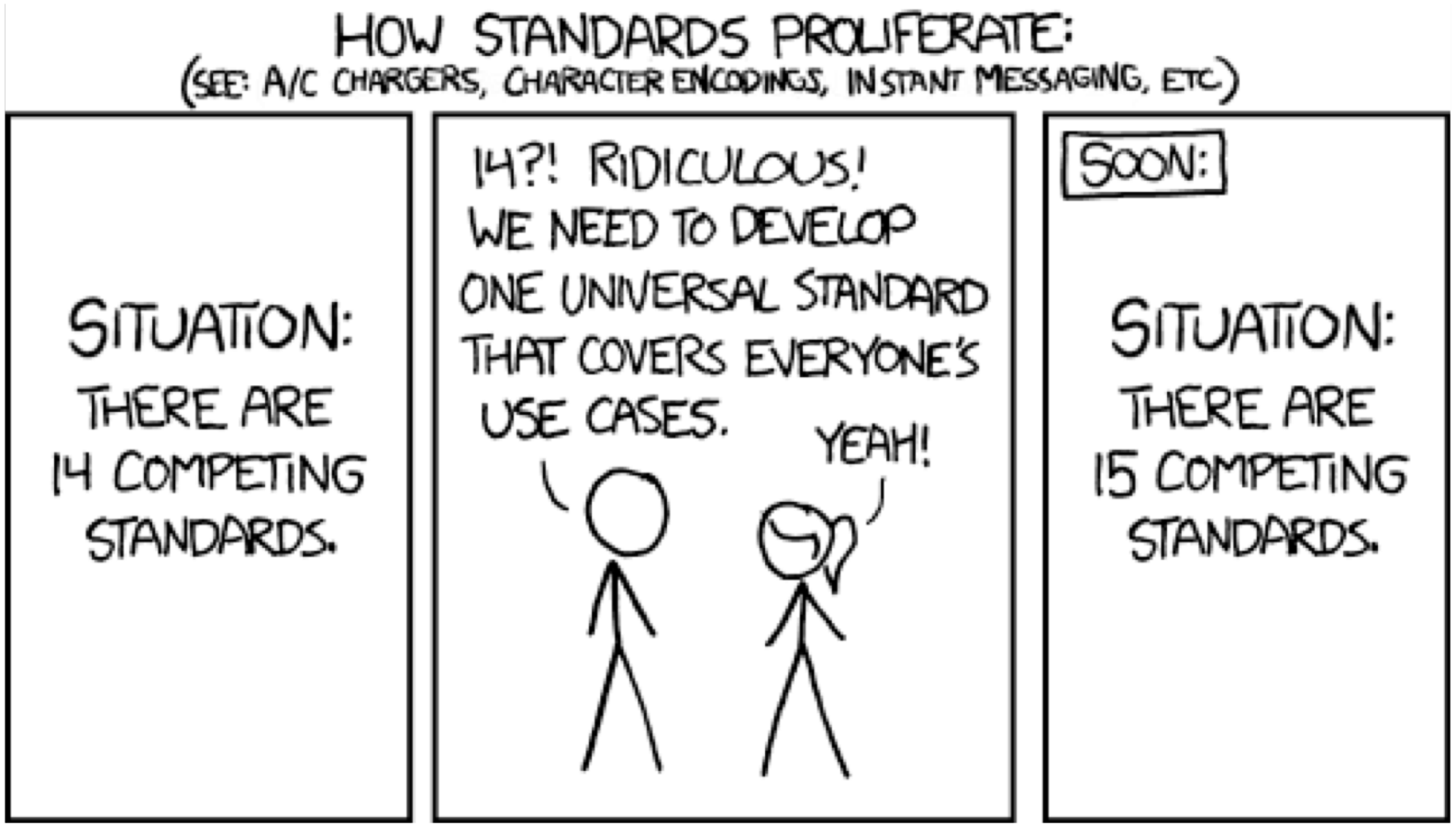
Structured Query Language
SQL - DDL

- SQL Overview
- SQL Datatypes
- DDL statements

SQL

- SQL is the query language for the **System R** developed at IBM San Jose
  [Astraham, Gray, Linsday, Selinger,…]
- SQL is the de-facto standard on most RDBMS
- Most successful standardization effort
  - SQL (ANSI 1986)
  - SQL1 (ANSI 1989)
  - SQL2 or SQL92 (ANSI 1992)
  - SQL3 (ANSI 1999/2000/2003) -- Core and Packages
  - SQL 2008
  - SQL 2013
  - SQL 2018

A word about Standards



http://xkcd.com/927/

Database Languages

- **Data Definition Language (*DDL*):**
  - Define schemas
  - Define **Integrity Constraints**
    - Example: unique *SID*s
  - More…
- **Data Manipulation Language (*DML*):**
  - To ask questions = ***Query***
    - Example: Which students have GPA > 3.75?
  - To insert, delete and update data

Basic SQL-DDL COMMANDS

- For database schemas:

  CREATE SCHEMA, DROP SCHEMA

- For tables:

  CREATE TABLE, DROP TABLE, ALTER TABLE

- For domains:

  CREATE DOMAIN, DROP DOMAIN [SQL99]

- For views:

  CREATE VIEW, DROP VIEW

- For integrity constraints

  CREATE IC, DROP IC

For Indexes [defunct in SQL2]

Create Database

- **CREATE DATABASE** <name>

  [[WITH] [OWNER [=] user_name]
  [TEMPLATE [=] template]
  [ENCODING [=] encoding]
  [LC_COLLATE [=] lc_collate]
  [LC_CTYPE [=] lc_ctype]
  [TABLESPACE [=] tablespace_name]
  [ALLOW_CONNECTIONS [=] allowconn]
  [CONNECTION LIMIT [=] connlimit]
  [IS_TEMPLATE [=] istemplate]]

- E.g., **CREATE DATABASE** cs1555 **OWNER** panos;

Create Database Schema

- A schema is essentially a namespace: it contains named objects (tables, data types, functions, and operators)
- **CREATE SCHEMA** <sc-name>
  AUTHORIZATION <user-identifier>;
- E.g. **CREATE SCHEMA** micro_db

  AUTHORIZATION panos;
- **DROP SCHEMA** <sc-name> [**RESTRICT** | **CASCADE**];
  - Restrict: removes the schema if the schema has no any objects
  - Cascade: removes everything, data and definitions
- E.g., **DROP SCHEMA** micro_db **RESTRICT**;

Schema and Catalog

- SQL2, SQL3 support multiple database schemas
- **Catalog** contains the definitions of database schemas
- INFORMATION_SCHEMA
  - Schemas and Base relations (tables)

    (tbl_name, creator, #of_tuples, tuple_length, #of_attributes…)
  - Attributes of Relations (columns)

    (tbl_name, atrb_name, type, format, order, key_no, ...)
  - Indexes

    (tbl_name, index_name, key_attribute,...)
  - Authorization
  - Integrity
- Naming of tables: Schema_name.Table_name
- Query: Describe table name; or using SELECT

Create Table

- **CREATE Table** <Table-name> **(**
  <Attribute-name> <Attribute-Type>, …
  Constraint <Constraint-name> <Constraint-spec>, …
  );

- E.g., **CREATE TABLE** Students (

  sid `CHAR(20),`

  name `CHAR(20),`

  psid `INTEGER,`

  age `INTEGER,`

  gpa `REAL,`

  Constraint Student_PK

  PRIMARY KEY (*sid*));

SQL Datatypes

- Numeric
  - Fixed numbers, approximate numbers, formatted numbers
- Character Strings
  - fixed & varying length, CLOBS [SQL99], foreign language
- Bit Strings
  - fixed & varying length, BLOBS [SQL99]
- Temporal Data
  - date, time and timestamp, intervals
- **NULL** value valid for all types

Constraints on Attributes

- Constraints:
  - NOT NULL
  - DEFAULT value
    - without the DEFAULT-clause, the default value is NULL
  - PRIMARY KEY (attribute-list)
  - UNIQUE (attribute list)
    - allows the specification of alternative key
  - FOREIGN KEY (key) REFERENCES table (key)

Create Table Schema - Example

```
CREATE TABLE STUDENT
  ( SID    INTEGER,
    Name   CHAR(20),
    PSID   INTEGER NOT NULL,  -- REQUIRED for AK
    AGE    INTEGER,
    GPA    REAL,
    Major  CHAR (10),
    CONSTRAINT STUDENT_PK
       PRIMARY KEY (SID),
    CONSTRAINT STUDENT_UN
       UNIQUE (PSID),
    CONSTRAINT STUDENT_FK
       FOREIGN KEY(Major) REFERENCES Department (DNO)
       ON UPDATE CASCADE ON DELETE NO ACTION
  );
```

Observations on Numeric types

- They are like the datatype in C
  - BIGINT for long integer or integer
- Truncation is towards 0
- Rounding is business instead of Scientific
  - [0..4] ↓ 0 [0..4] ↓ 0
  - [6..9] ↑ 1 [5..9] ↑ 1
  - Half times of 5 is 0 and half 1

- *Money* or C*urrency* data are numeric data with a currency sign: $, £, €, ¥

Date and Time

- DATE (10 positions) stores calendar values representing YEAR, MONTH, and DAY: **YYYY-MM-DD**
- TIME defines HOURS, MINUTES, and SECONDS in a twenty-four-hour notation: **HH:MM:SS**
- TIME(i) defines *i* additional decimal fractions of seconds: **HH:MM:SS:ddd...d**
- TIME WITH TIME ZONE includes the displacement [+13:00 to -12:59] from standard universal time zone: **HH:MM:SS{+/-}hh:mm**
  - *hh* are the two digits for the TIMEZONE_HOUR and *mm* the two digits for TIMEZONE_MINUTE
- TIMESTAMP represents a complete date and time with 6 fractions of seconds and optional time zone.

Functions on Dates

- All systems provide functions under different names
  - for constructing a date from strings or integers
  - for extracting out the month, day, or year from a date
  - for displaying dates in different ways
- Examples
  - CAST(string AS DATE) [SQL2: CAST(<value> AS <type>)]

    e.g., CAST('2002-02-18' AS DATE)
  - MAKEDATE (int year, int month, int day) or
    DATE (int year, int month, int day)

    e.g., MAKEDATE(1999, 12, 31)
  - EXTRACT (MONTH/DAY/YEAR FROM <date>) [SQL3]

    e.g., EXTRACT (month from DATE(2020, 9, 29))
  - YEAR(<date>), MONTH(<date>), DAY(<date>)

Constructing Date Functions in PSQL

Functions	Returns
TO_CHAR(d,format)	character-string equivalent of *d* based on *format*
TO_DATE(s,format)	date corresponding to *s* based on *format*
TO_TIMESTAMP(s, format)	date corresponding to *s* based on *format*

Examples:

- TO_DATE('2011-FEB-18', 'YYYY-MON-DD')
- TO_DATE('02182011' , 'MMDDYYYY')
- TO_CHAR(mydate, DY) → returns sun, mon, tue, wed, thu, fri, sat

Format	Description
MM	Month number
MON	3-letter abbreviation of month
MONTH	Fully spelled-out month
D	Number of days in the week
DD	Number of days in the month
DDD	Number of days in the year
DY	3-letter abbreviation of day of week
DAY	Fully spelled-out day of week
Y, YY, YYY, YYYY	Last 1, 2, 3 or 4 digits of year
HH12, HH24	Hours of the day (1-12 or 0-23)
MI	Minutes of hour
SS	Seconds of minute
AM, PM	Display AM or PM depending on time

Resolving Spec Ambiguity

- TO_DATE(‘02182011’ , ‘MMDDYYYY’)
- It parses to the longest keyword.

- Examples:
  - 'DYY' = DY and Y

    TO_DATE('WED7', 'DYY') = 01-FEB-17
  - 'DDDYYYY' = DDD and YYYY

    TO_DATE('3232017', 'DDDYYYY') = 19-NOV-17
  - 'DYYY' = DY and YY

    TO_DATE('WED17', 'DYYY') = 01-FEB-17

Intervals

- An interval results when two dates are subtracted. E.g., AdmitDate – DischargeDate
- Two interval data types: **Year-Month** & **Day-Time**
- Format: INTERVAL start-field(p) [TO end-field(fs)]
  - p is the precision (default is 2 digits)
  - fs is the fractional second precision, which is only applicable to DAY/TIME (default is 6 digits)
- Year-Month intervals:
  - INTERVAL YEAR, INTERVAL YEAR(p), INTERVAL MONTH, INTERVAL MONTH(p), INTERVAL YEAR TO MONTH, INTERVAL YEAR(p) TO MONTH
  - INTERVAL YEAR (2) to MONTH could be [0-0, 99-11]
  - INTERVAL YEAR TO MONTH '123-04' is 123 years, 4 months