"Bacon" SQL features

These features are in the *.sql* files on *.../bacon-numbers-slides-and-companion-code/bacon-numbers/* wherever you unzipped *bacon-numbers-slides-and-companion-code.zip*.

1. Classic three-table implementation of a many-to-many actors-plays ERD relationship — *01-cr-tables.sql*

Uses *cast_members* intersection table with, for example:

```
cast_members_fk1 foreign key(actor)
  references actors(actor)
  match full
  on delete cascade
  on update restrict
```

2. Three-table implementation of a many-to-many actors-actors ERD relationship — *02-cr-edges-table-and-proc.sql*

Uses *edges* intersection table with with an *array* attribute: *movies text[]*.

3. Language *plpgsql* procedure to populate the edges table — *02-cr-edges-table-and-proc.sql*

Uses inner self-join on *edges* and *select... array_agg(movie order by movie)* to populate *movies[]*. Encapsulates *delete* and *insert* as atomic business txn.

4. Language *plpgsql* procedure using dynamic SQL DDLs — *05-cr-raw-paths-table.sql*

Creates table with run-time-specified name with standard column structure. Anciliary objects (like a *sequence*) have systematically related created names.

5. Language *plpgsql* table function — *06-cr-list-paths.sql*

Creates an easy-to-read list of paths where one path is displayed as a sequence of traversed nodes. Uses *translate()* built-in function to render the *::text* typecast of movies array more readable *declaratively*.

6. Use of a recursive CTE to find all the distinct paths in an undirected cyclic graph (encapsulated in language *plpgsql* proc) — *07-find-paths-naive.sql*

Shows how to build the paths starting with the *array constructor* in the non-recursive term and *array concatenation* in the recursive term. Uses *plpgsql* procedure *terminal(some_array)* to encapsulate *some_array[cardinality(some_array)]* to get the last element in an array. Uses the array to prevent cycles by comparing the to-be-added element with every element on the path to date using *any(some_array)*. The procedure encapsulates *drop index*, *delete* from to-be-populated table, and *insert* into it.

7. Use of *after insert* trigger on paths to populate a column that traces the execution — *08-cr-raw-paths-with-tracing.sql*

A common technique for use at development time to add execution tracing without cluttering the main code.

8. Use of a language *plpgsql* procedure to implement recursive CTE algorithm as SQL-PL/pgSQL hybrid — *09-find-paths-no-pruning.sql*

This is a well-known technique that database developes come to rely on when the use-case can't be met with a singe SQL statement.

9. Use of the @> ("contains") array operator — *10-cr-restrict-to-unq_containing-paths.sql*

This is used to remove shorter paths that are completely contained by longer paths. It's hard to make up a convincing stand-along demo that shows the value of this operator. Here, it's the other way round. There's a clear problem to be solved. And "contains" meets the need.

10. Use of the *\copy* metacommand to ingest data from an o/s file — *13-insert-imdb-data.sql*

Used to ingest real IMBd — a curated subset from a University site that represents a fully connected graph with so many connected actors and movies (maximum Bacon number is six) that the pure SQL approach never finishes. Shows that the SQL-PL/pgSQL hybrid is is *essential*.

"Employee hierarchy" SQL features

This adds these features that are not used in the "Bacon" use case. They are in the *.sql* files on *.../bacon-numbers-slides-and-companion-code/employee-hierarchy/* wherever you unzipped *bacon-numbers-slides-and-companion-code.zip*.

11. Creates *domain name_t as text* — *1-cr-table.sql*

This is used for the *emps.name* and *emps.mgr_name* columns. Constrains the max lenth to 30 characters. Much easier to change than if the coulmn were simply *varchar(30)*. Also enforces the rule that names are all lower case.

12. Unique expression-based, partial index — *1-cr-table.sql*

Enforces the business rule that there is no more than one ultimate manager.

13. Self-referential FK constraint (a.k.a. pig's ear) — *1-cr-table.sql*

Together with the unique expression-based, partial index, this enforces the rule that every employee except for the ultimate manager must have exactly one manager—i.e. that the data is a strict rooted tree (a.k.a. hierarchy).

14. Use of *order by... nulls first* — *1-cr-table.sql*

Lists employees in manager order with the ultimate manager first.