Session 3: Relational Data Model (CH-3)
CSCI-585, Cyrus Shahabi

- Relational data model represents the database as a collection of tables, termed *relations*, each with a unique name (**not to be confused** with the *relationships* in ER model).

- A row (also termed a *tuple* or a record) in a table represents a collection of related values.
- Each value corresponds to an *attribute* or a field.
- A column of a table represents an attribute.
- Since a table is a collection of such relationships, the concept of table is very similar to the mathematical concept of relation.
- Given a relation with attributes (A_1:D_1, A_2:D_2, …, A_n:D_n) the domain of this relation is D_1 x D_2 x … x D_n. The relation (or table) itself is a subset of this domain (e.g., student (SS#: integer, name: string, gpa: float) --- domain: integer x string x float)
- Database *scheme* describes the logical design of the database (e.g., the table frame).
- Database *instance* is the data in the database at a given instant in time (e.g., the contents of the table).

Various types of constraints:

- **Domain constraints:** Value of each attribute A must be an atomic value from the domain for that attribute (e.g., age: integer; 4 is acceptable and 4.3 is not!).

- **Key constraints:** Since a relation is a **set** of tuples, by definition all tuples in a relation must also be distinct. That is, no two tuples can have the same combination of values for *all* their attributes. Same concepts of superkey, candidate key and primary key apply here.
- **Entity integrity constraint:** No primary key value can be null.

(Issue: Different attribute names refer to the same concept in real world and cannot be forced to have the same name. Meanwhile, same attribute names might refer to different concepts in real world. Hence, some sort of regulation (or constraint) required.)

- **Referential integrity constraint:** A tuple in one relation that refers to another relation must refer to an *existing tuple* in that relation (e.g., value of DNO for EMPLOYEE must match the Dnumber value of some tuple in DEPARTMENT).
    - Set of attributes FK in R1 is *foreign key* of R1 if:
        1. Attributes in FK have the same domain as the primary key attributes PK of relation R2.
        2. A value of FK in a tuple t1 of R1 either occurs as a value of PK for some tuple t2 in R2 or is null.

- A foreign key can refer to its own relation (e.g., manager_ss#).

ER-to-Relational Mapping

- Strong entity set with attributes a_1, a_2, …, a_n: represent it as a table with *n* unique columns (one column per attribute).
  Example: ….

  Each row in this table corresponds to one entity of the entity set.
  We may add/delete/modify rows in the table.
- Weak entity set with attributes a_1, a_2, …, a_n and an owner entity set with primary key b_1, b_2, …, b_m : represent it as a table with n+m columns, one for each of { a_1, a_2, …, a_n} U { b_1, b_2, …, b_m }. b_1, b_2, …, b_m is the foreign key of the resulting relation referring to the corresponding relation of the owner entity set.
  Example:

- *(Idea: keep rows unique.)*

- N-ary relationship set R with attributes a_1, a_2, …, a_n among entity sets E_i 's (say m entity sets): represent it as a table with n+m columns, one for each of { a_1, a_2, …, a_n} U {prim-key(E_1), prim-key(E_2), …, prim-key(E_m)}.
- Binary relationship set R with attributes a_1, a_2, …, a_n among entity sets corresponding to relations S and T:
  - If 1:1 then choose either relations (say S) and extend it with prim-key(T) U { a_1, a_2, …, a_n}
  - If 1:N or N:1 then choose the N-side relation (say S) and extend it with prim-key(T) U { a_1, a_2, …, a_n}
  - If N:M then create a new relation as:
    prim-key(S) U prim-key(T) U { a_1, a_2, …, a_n}

- For multivalued attribute A of entity set S, create a new relation as: A U prim-key(S)

How to enforce 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?
  - Also delete all Enrolled tuples that refer to it.
  - Disallow deletion of a Students tuple that is referred to.
  - Set sid in Enrolled tuples that refer to it to a *default sid* .
  - (In SQL, also: 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.

SQL/ 92 supports 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),
cid CHAR( 20) ,
grade CHAR (2),
PRIMARY KEY (sid, cid),
FOREIGN KEY (sid)
REFERENCES Students
ON DELETE CASCADE
ON UPDATE SET DEFAULT )
```