8.2: *n*-ary Relations

Rosen 6th ed., Ch. 8

§7.2: *n*-ary Relations

- An *n*-ary relation R on sets $A_1,\ldots,A_n$, written (with signature) $R{:}A_1\times\ldots\times A_n$ or $R{:}A_1,\ldots,A_n$, is simply a subset
$$R \subseteq A_1 \times \ldots \times A_n.$$
- The sets A_i are called the *domains* of R.
- The *degree* of R is n.
- R is *functional in the domain* A_i if it contains at most one n-tuple $(\ldots, a_i, \ldots)$ for any value a_i within domain A_i.

Relational Databases

- A *relational database* is essentially just an n-ary relation R.
- A domain A_i is a *primary key* when the value of the n-tuple from this domain determines the n-tuple. That is, a domain is a primary key when no two n-tuples in the relation have the same value from this domain.

Relational Databases

- A *composite key* for the database is a set of domains $\{A_i, A_j, \ldots\}$ such that R contains at most 1 n-tuple $(\ldots,a_i,\ldots,a_j,\ldots)$ for each composite value $(a_i, a_j,\ldots)\in A_i \times A_j \times \ldots$

Let's create an example

- Example: Student records

Database Tables

- Relations that represent databases are also called **Tables**.
- Why? Well, these relations are usually displayed as tables.
  - 1 row for each tuple in the relation
  - N-columns in the table for an N-tuple

Operations on n-ary relations

- Database queries:
  - Operations that form new n-ary relations from n-ary relations
  - To find sets of records from a database that match some condition (selection)
  - To delete the same fields in every record of a relation (projection)
  - To combine multiple tables when they share one or more identical fields (join)

Selection Operators

- Let A be any n-ary domain $A=A_1\times\ldots\times A_n$, and let $C:A\rightarrow\{\mathbf{T},\mathbf{F}\}$ be any *condition* (predicate) on elements (n-tuples) of A.
- Then, the *selection operator* s_C is the operator that maps any (n-ary) relation R on A to the n-ary relation of all n-tuples from R that satisfy C.
  - *I.e.*, $\forall R\subseteq A,\ s_C(R) = \{a\in R \mid s_C(a) = \mathbf{T}\}$

Selection Operator Example

- Suppose we have a domain
  A = StudentName × Standing × SocSecNos
- Suppose we define a certain condition on A,
  $UpperLevel(name,standing,ssn) :\equiv$
  $[(standing = \text{junior}) \vee (standing = \text{senior})]$
- Then, $s_{UpperLevel}$ is the selection operator that takes any relation R on A (database of students) and produces a relation consisting of *just* the upper-level classes (juniors and seniors).

Projection Operators

- Let $A = A_1 \times \ldots \times A_n$ be any n-ary domain, and let $(i_1,\ldots,i_m)$ be a sequence of indices all falling in the range 1 to n,
  - That is, where $1 \le i_k \le n$ for all $1 \le k \le m$.
- Then the *projection operator* on n-tuples is defined by:

$$P_{(i_1,\ldots,i_m)} : A \to A_{i_1} \times \ldots \times A_{i_m}$$

$$P_{(i_1,\ldots,i_m)}(a_1,\ldots,a_n) = (a_{i_1},\ldots,a_{i_m})$$

Projection Example

- Suppose we have a ternary (3-ary) domain $Cars = Model \times Year \times Color$. (note $n=3$).
- Consider the index sequence $(1,3)$. ($m=2$)
- Then the projection $P_{(1,3)}$ simply maps each tuple $(a_1,a_2,a_3) = (model,year,color)$ to its image:

$$(a_{i_1}, a_{i_2}) = (a_1, a_3) = (model, color)$$

- This operator can be usefully applied to a whole relation $R \subseteq Cars$ (a database of cars) to obtain a list of the model/color combinations available.

Join Operator

- Puts two relations together to form a sort of combined relation.
- If the tuple (A,B) appears in R_1, and the tuple (B,C) appears in R_2, then the tuple (A,B,C) appears in the join $J(R_1,R_2)$.
  - A, B, and C here can also be sequences of elements (across multiple fields), not just single elements.

Join Example

- Suppose R_1 is a teaching assignment table, relating *Professors* to *Courses*.
- Suppose R_2 is a room assignment table relating *Courses* to *Rooms*,*Times*.
- Then $J(R_1,R_2)$ is like your class schedule, listing (*professor*,*course*,*room*,*time*).

Structured Query Language (SQL)

- Database query language
- Used to carry out operations that have been described
- Queries can be made on multiple tables, multiple fields, etc.
- Note: SQL uses the SELECT command as a *projection* rather than a *selection* operator

SQL Examples

- Imagine we have a table of flight records

```
SELECT Departure_time
FROM Flights
WHERE Destination='Detroit'
```

- Above would give a list of departure times (only) and no other fields
- Really a projection (sort of a combo of selection and projection actually)

SQL Examples

- Example involving a join:
- Imagine we have a college information system database with tables for teaching assignments (which prof teaching which course) and class schedule (when and where classes meet)

```
SELECT Professor, Time
FROM Teaching_assignments, Class_schedule
WHERE Department='Mathematics'
```

- The FROM is specifying a join, the SELECT is specifying a projection of the result of that join.