Modification of Database

Deletion

- A query and an assignment
  - r ← r – E
- Only delete tuples not individual attribute values

Insertion

- Insert either:
    - a tuple
    - an expression that gives a set of tuples
- r ← r U E
- Value of attributes of inserted tuple must be member of attribute’s domain

Updating

- Only change some values in a tuple
- Use generalized projection
- $r \leftarrow \pi_{F1, F2, \ldots, Fn}(r)$ (updates all tuples)
  - Fi is the attribute of r (if not to be changed) or an expression involving the attribute
- To update only some tuples
  - $r \leftarrow \pi_{F1, F2, \ldots, Fn}(\sigma_P(r)) \cup (r - \sigma_P(r))$

Chapter 3: SQL

SQL

- A user-friendly *query language*, unlike relational algebra
- SQL uses relational algebra + calculus
- More than just a query language
    - define data structure (schema)
    - modify data in DB
    - specify constraints
- Implementations may differ

Background

- IBM developed 1st version in 70s, Sequel
    - Structured English Query Language
    - call it S-Q-L
- Parts
    - **DDL**. defining relation schema, deleting relations, modifying relation schema
    - **Interactive DML**. QL based on rel algebra + calculus
        - also insert, delete, and modify tuples
    - **Specifying Integrity constraints**. updates violating these will be disallowed
    - **Views**. Allows view defnitions
    - **Transaction specs**. start & end of transactions
    - **Embedded SQL.**
    - **Authorization**. access rights for relations & views

Sample relations

- employee(name, dept_id, cat_id)
- department(dept_id, dept_name, floor)
- category(cat_id, cat_name)
- project(pr_name, manager_id, assets)

  - This notation omits the schema name

Data Definition

- Schema (attributes) for relation
- domain of values for each attribute
- integrity constraints
- indices for each relation
- security and authorization
- physical storage structure of each relation

Basic domains

- `char(n)`. fixed-length, blank characters for filler upto n
- `varchar(n)`. upto n. No filler characters. allocated in increments
- `int`. machine-dependent
- `smallint`. small integer. machine-dependent
- `numeric(p, d)`. fixed-point no. with user-defined precision. p: total digits, d: after decimal
- `real` and `double`. machine-dependent
- `float(n)`. user-defined precision till n digits
- some more, e.g. `date`, covered later

Schema definitions in SQL

create table *r(*A_1 D_1, A_2 D_2,…, A_1 D_1,
<integrity_constraint$_1$,
integrity_constraint$_2$,
……
integrity_constraint$_k$>)

r: name of relation
A_i: name of attribute
D_i: domain of attribute

Will only consider one constraint: **primary keys**

- **primary key (A_j, A_k, …, A_m).** non-null and unique values. otherwise updates are rejected

Creating the tables for our schems

```
·create table employee
       (name char(20),
        dept_id int(10),
        cat_id int(10),
        primary key (name))

·create table department
       (dept_id int(10),
        dept_name char(20),
        floor int(10),
        primary key (dept_id))

·create table category
       (cat_id int(10),
        cat_name char(20),
        primary key (cat_id))

·create table project
       (pr_name char(20),
        manager_id char(10),
              assets numeric(10, 2),
        primary key (pr_name))
```