Database Design

October 27, 2008

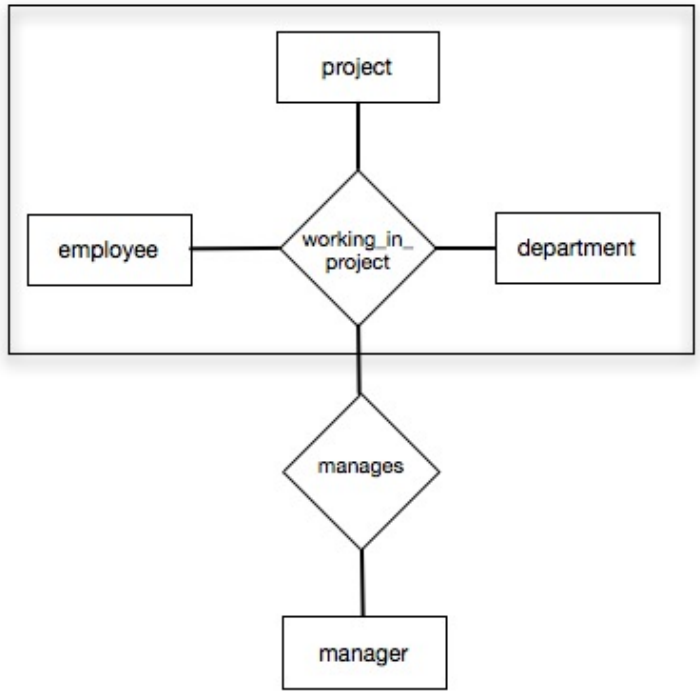
Composite and Multi–Valued Attributes

- Composite attribute is expanded into multiple attributes
  - original attribute is discarded
- New relation is created for a multi–valued attribute
- If M is multi–valued:
  - New relation R is created
  - Attributes
    1. A: same as M
    2. primary keys of M's entity set (act as foreign key)
  - Primary key $\rightarrow$ all attributes
  - Create foreign key via shared attribute

Representing Generalization in Relation Schema

- Two ways; both have merits
- First:
  - Schema for higher-level ES
  - Schema for lower ESs with attributes:
    - extra attributes for specialization
    - primary keys of higher–level ES
  - Primary–key is the PK of higher–level ES
  - Foreign key is the shared PK

Representing Aggregation in Relation Schema



- Schema $manages_s$ has attributes:
  - PK(*manager*)
  - PK(*works_in_project*)
- The rest are translated according to standard rules

Relational Schema for Bank

- Schemas from strong entity:
  $branch = (\underline{branch\_name}, branch\_city, assets)$
  $customer = (\underline{customer\_id}, customer\_name, customer\_street, customer\_city)$
  $loan = (\underline{loan\_number}, amount)$
  $account = (\underline{account\_number}, balance)$
  $employee = (\underline{employee\_id}, employee\_name, telephone\_number, start\_date)$
- Schemas from multivalued attributes:
  $dependent\_name = (\underline{employee\_id, d\_name})$

Relational Schema continued

- Schemas from relationship sets between strong ES:
  account_branch = (account_number, branch_name)
  loan_branch = (loan_number, branch_name)
  borrower = (customer_id, loan_number)
  depositor = (customer_id, account_number)
  cust_banker = (customer_id, employee_id, type)
  works_for = (worker_employee_id, manager_employee_id)
- Schemas from weak entity set
  payment = (loan_number, payment_number, payment_date, payment_amount)
- Schemas from ISA relationships
  savings_account = (account_number, interest_rate)
  checking_account = (account_number, overdraft_amoount)

Outline

Data Constraints

- Primary key, foreign key, check, assertions, triggers
- Automate consistency preservation
  - More reliable than application logic
  - Central location for constraints
- Also aid in physical structure of data
  - Store related data in proximity
  - Also indexes are better on primary key
- Comes at a price in performance
  - Faulty updates must be rejected
  - Performance depends upon DB design

Usage Requirements: Queries, Performance

- Computing resources (software+hardware),
- **people** and **external processes** efficiency
- Metrics
  - **Throughput**: queries/updates (*transactions*) per unit time
  - **Response Time**: time for a single transaction (avg. or worst case)
- High throughput: focus of batch style DBs
  - Most commercial DBs
  - High utilization of resources; may delay some transactions (greater good)
- High response: people–oriented or real–time systems
- Web-based, telecommunication ISs