Principles of DB Management and Use
CS743
Fall 2014

Database Management

Basic idea

- Remove details related to data storage and access from application programs.
- Concentrate those functions in single subsystem: the **Database Management System** (DBMS).
- Have all applications access data through the DBMS.

Advantages

- Uncontrolled redundancy can be reduced.
- Risk of inconsistency can be reduced.
- Data integrity can be maintained.
- Access restrictions can be applied.
- Physical data independence for programs

The Three-Schema Architecture

A **schema** describes the structure of the data in terms of some data model.

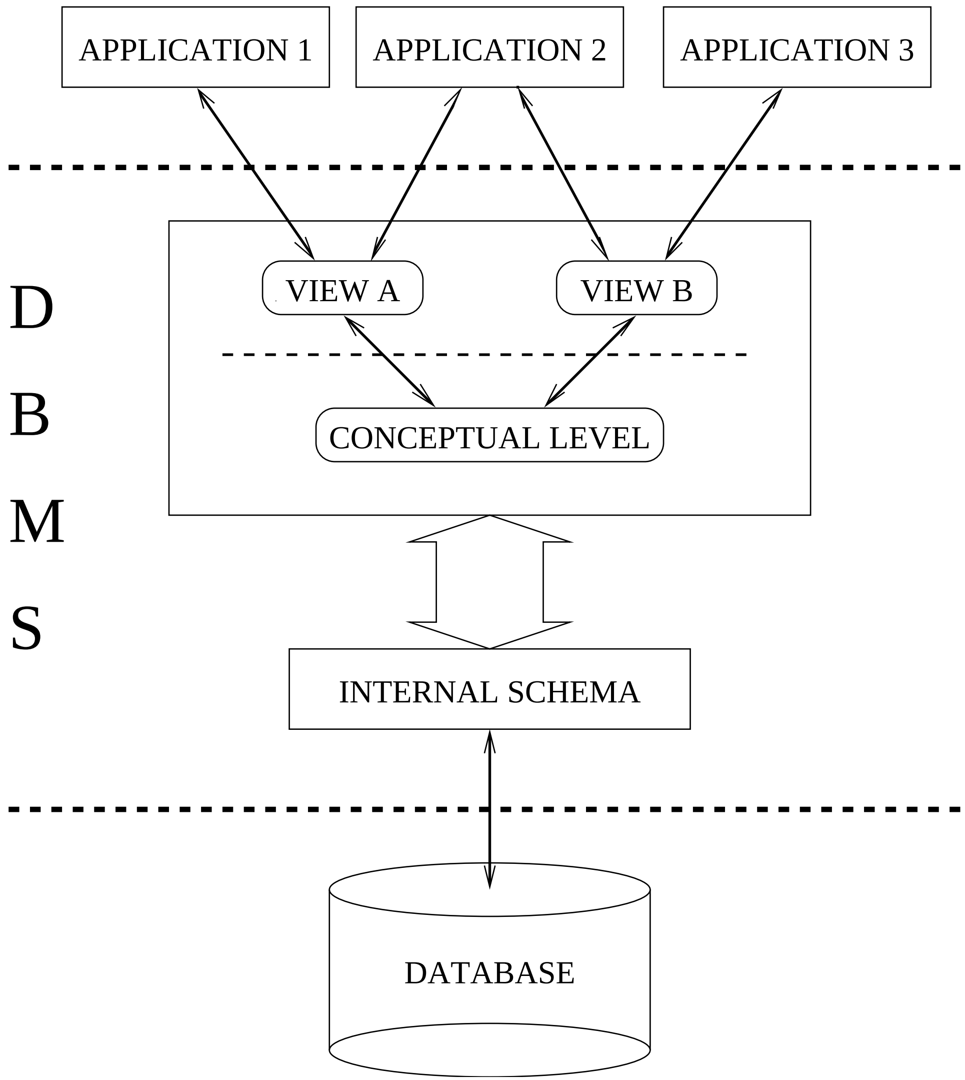
- External schema (view): describes data as seen by an application program
- Conceptual schema: describes the logical structure of all data
- Internal schema: describes how the database is physically encoded

Separation of external schema from conceptual schema enables logical data independence.

Separation of conceptual schema from internal schema enables physical data independence.

A database schema is different from a database instance.

The Three-Schema Architecture (cont'd)



Interfacing to the DBMS

Data Definition Language (DDL): for specifying schemas

- may have different DDLs for external schema, conceptual schema, internal schema
- information is stored in the **data dictionary**, or **catalog**

Data Manipulation Language (DML): for specifying queries and updates

- **navigational** (procedural)
- **non-navigational** (declarative)

The Relational Data Model

- a database is a set of uniquely named relations
- a relation is a set of tuples
  - each relation has a fixed set of uniquely named attributes
  - in addition to its name, each attribute has an associated domain
  - a domain is a set of values
  - every tuple in a relation is a set of values, one value from the domain of each of that relation's attributes

Attribute values must be **atomic**: no tuples or sets or . . ..

A Relation

Department

DeptNo	DeptName	MgrNo	AdmrDept
A00	Planning	000020	A00
E01	Support Services	000050	A00
E11	Operations	000090	E01
E21	Software Support	000100	E01

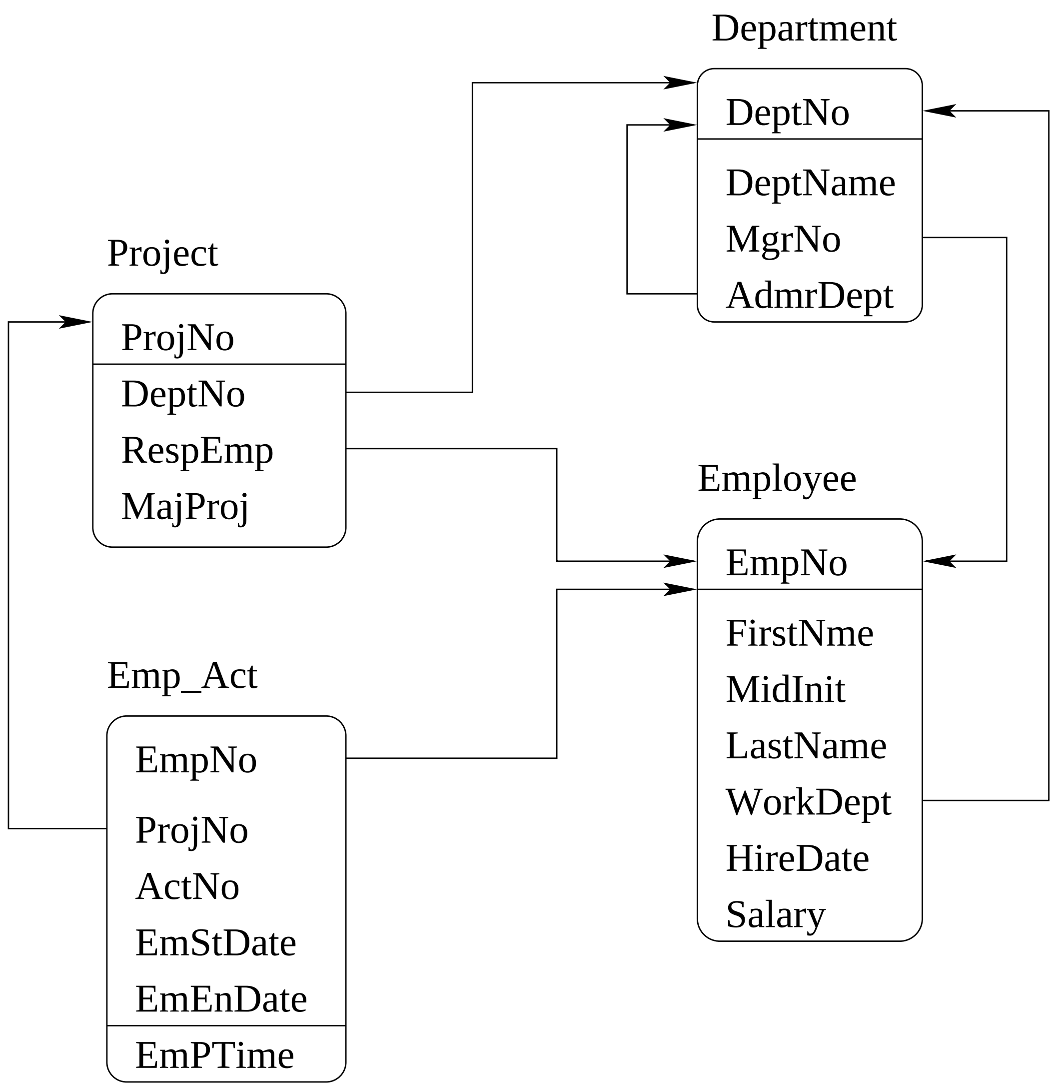
Relation Schema

- The schema of a relational database includes the schemas of its relations.
- The schema of a relation includes the relation's name, the names of its attributes, and their associated domains.
  - A schema usually includes additional information about the logical structure of the data, such as key constraints.
  - A relation's schema does not include the relation's tuples.

Constraints

- a *constraint* is a rule that restricts the tuples that may appear in a database instance
- common examples: primary key constraints, foreign key constraints
  - a primary key constraint for a relation R specifies a set of attributes of R whose values can be used to uniquely identify any tuple in R, i.e., no two tuples in R can have the same values for the key attribute(s)
  - a foreign key constraint specifies that values found in foreign key columns in a *referencing* relation R_1 must appear as primary keys in a *referenced* relation R_2

A Portion of the Schema for the DB2 Sample Database



Relational Algebra

- the relational algebra consists of a set of *operators*
- each operator operates on one or more relations
- each operator defines a single output relation in terms of its input relation(s)
- relational operators can be composed to form expressions that define new relations in terms of existing relations.

Some Relational Operators

- Selection ($\sigma_{condition}(R)$)
  - result schema is the same as R's
  - result relation includes a subset of the tuples of R
- Projection ($\pi_{attributes}(R)$)
  - result schema includes only the specified attributes
  - result relation would have as many tuples as R, except that duplicates are eliminated
- Product ($R \times S$)
  - result schema has all of the attributes of R and all of the attributes of S
  - result relation includes one tuple for every pair of tuples (one from each relation) in R and S
  - sometimes called cross-product or Cartesian product

Cross Product Example

R

AAA	BBB
a_1	b_1
a_2	b_2
a_3	b_3

S

CCC	DDD
c_1	d_1
c_2	d_2

$R \times S$

AAA	BBB	CCC	DDD
a_1	b_1	c_1	d_1
a_2	b_2	c_1	d_1
a_3	b_3	c_1	d_1
a_1	b_1	c_2	d_2
a_2	b_2	c_2	d_2
a_3	b_3	c_2	d_2

Select,Project,Product Examples

- Find the last names and hire dates of employees who make more than $100000.

$$\pi_{LastName,HireDate}(\sigma_{Salary>100000}(E))$$

- For each project for which department E21 is responsible, find the name of the employee in charge of that project.

$$\pi_{Name,LastName}(\sigma_{DeptNo=E21}(\sigma_{RespEmp=EmpNo}(E \times P)))$$

- Note: E is the Employee relation, P is the project relation
- division operator: inverse of product: $(A \times B)/B = A$
- this gives projects on which all employees participate

$$(\pi_{Projno,Empno}(Emp\_Act))/(\pi_{Empno}(Employee)$$

Joins

- Natural join ($R \bowtie S$) is a very commonly used operator which can be defined in terms of selection, projection, and Cartesian product.
- The result of $R \bowtie S$ can be formed by the following steps
  1. form the cross-product of R and S
  2. eliminate from the cross product any tuples that do not have matching values for all pair of attributes common to R and S
  3. eliminate any duplicate attributes
- Natural join is special case of *equijoin*, a common and important operation.

$$P \bowtie_{(\text{RespEmp}=\text{EmpNo})} E$$

Example: Natural Join

- Consider the natural join of the Project and Department tables, which have attribute DeptNo in common
  - the schema of the result will include attributes ProjName, DeptNo, RespEmp, MajProj, DeptName, MgrNo, and AdmrDept
  - the resulting relation will include one tuple for each tuple in the Project relation (why?)

Set-Based Relational Operators

- Union ($R \cup S$):
  - schemas of R and S must be "union compatible"
  - result includes all tuples that appear either in R or in S or in both
- Intersection ($R \cap S$):
  - schemas of R and S must be "union compatible"
  - result includes all tuples that appear in both R and S
- Difference ($R - S$):
  - schemas of R and S must be "union compatible"
  - result includes all tuples that appear in R and that do not appear in S

Relational Division

S

B	C
b_1	c_1
b_1	c_2
b_2	c_2

X

A	B	C
a_1	b_1	c_1
a_1	b_1	c_2
a_1	b_2	c_2
a_2	b_1	c_1
a_2	b_1	c_2
a_2	b_2	c_2

X/S

A
a_1
a_2

Division is the Inverse of Product

R

A
a_1
a_2

S

B	C
b_1	c_1
b_1	c_2
b_2	c_2

$R \times S$

A	B	C
a_1	b_1	c_1
a_1	b_1	c_2
a_1	b_2	c_2
a_2	b_1	c_1
a_2	b_1	c_2
a_2	b_2	c_2

$(R \times S)/S$

A
a_1
a_2

Algebraic Equivalences

- This:

$$\pi_{Name,LastName}(\sigma_{DeptNo=E21}(\sigma_{RespEmp=EmpNo}(E \times P)))$$

- is equivalent to this:

$$\pi_{Name,LastName}(\sigma_{DeptNo=E21}(E \bowtie_{RespEmp=EmpNo} P))$$

- is equivalent to this:

$$\pi_{Name,LastName}(E \bowtie_{RespEmp=EmpNo} \sigma_{DeptNo=E21}(P))$$

- is equivalent to this:

$$\begin{aligned}\pi_{Name,LastName}(\quad & (\quad \pi_{Name,LastName,Empno}(E)) \bowtie_{RespEmp=EmpNo} \\ & (\quad \pi_{RespEmp}(\sigma_{DeptNo=E21}(P))))\end{aligned}$$

- More on this topic later ...