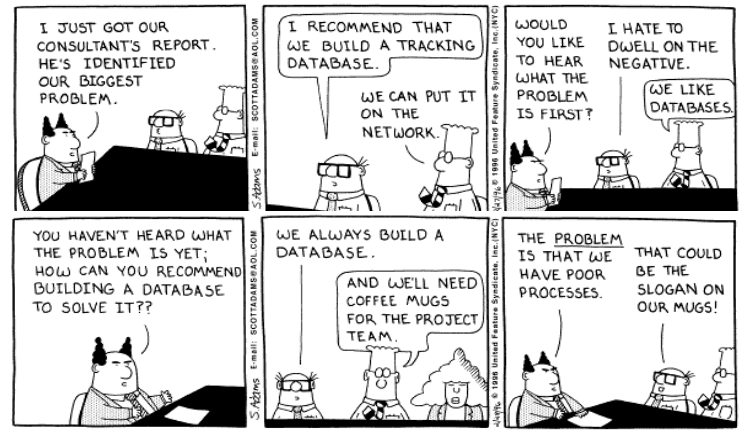
CS5208: Foundations on Database Systems (http://www.comp.nus.edu.sg/~cs5208)



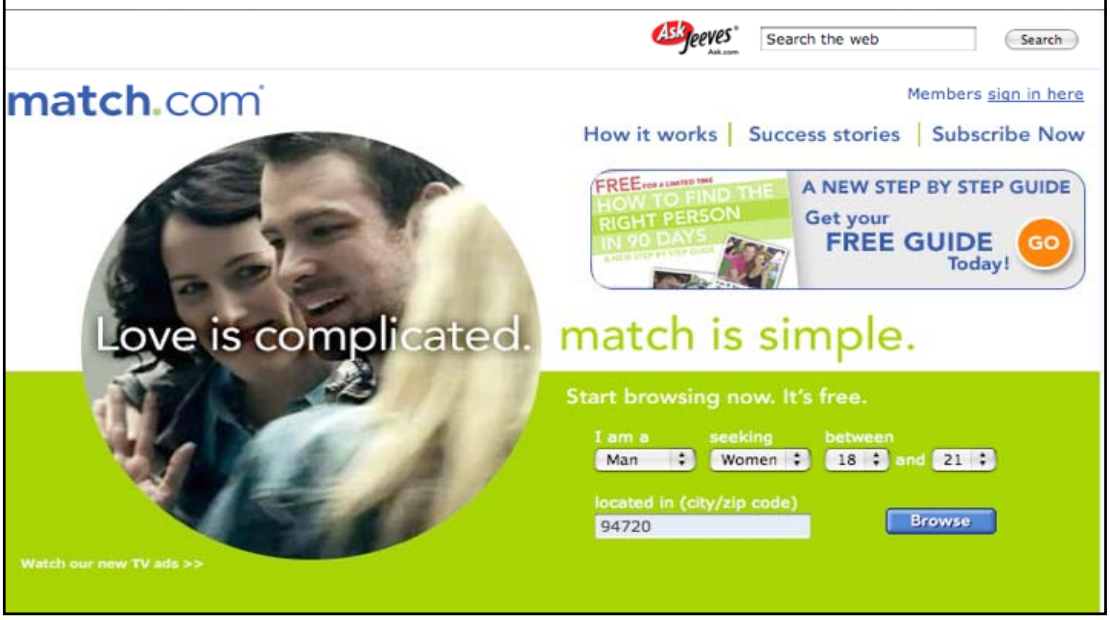
"Knowledge is of two kinds: we know a subject ourselves, or we know where we can find information upon it."

-- Samuel Johnson (1709-1784)

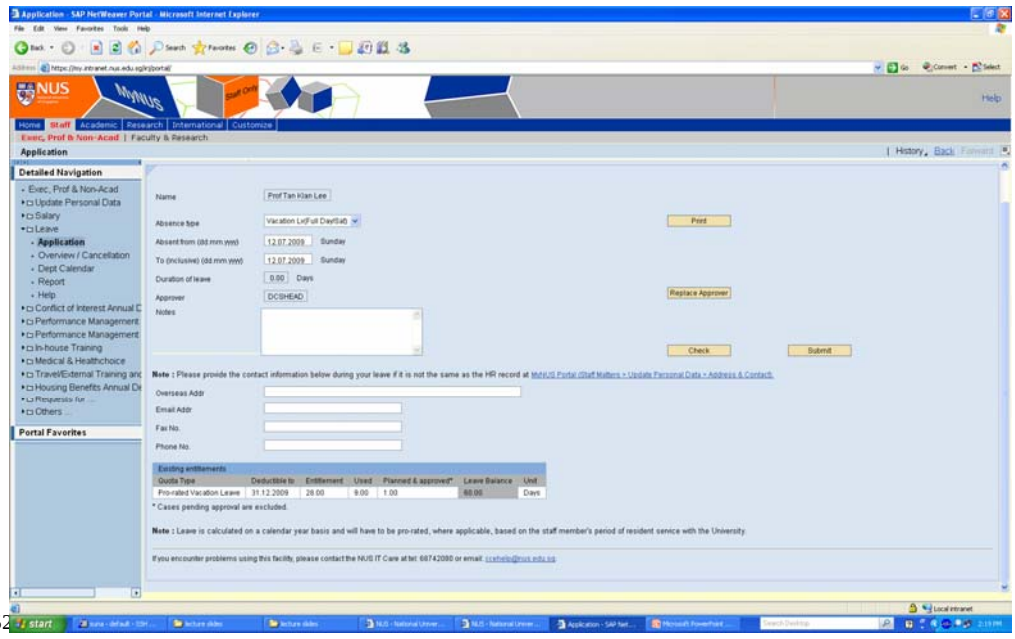
Lecture 1

Introduction & Data Design and Modeling

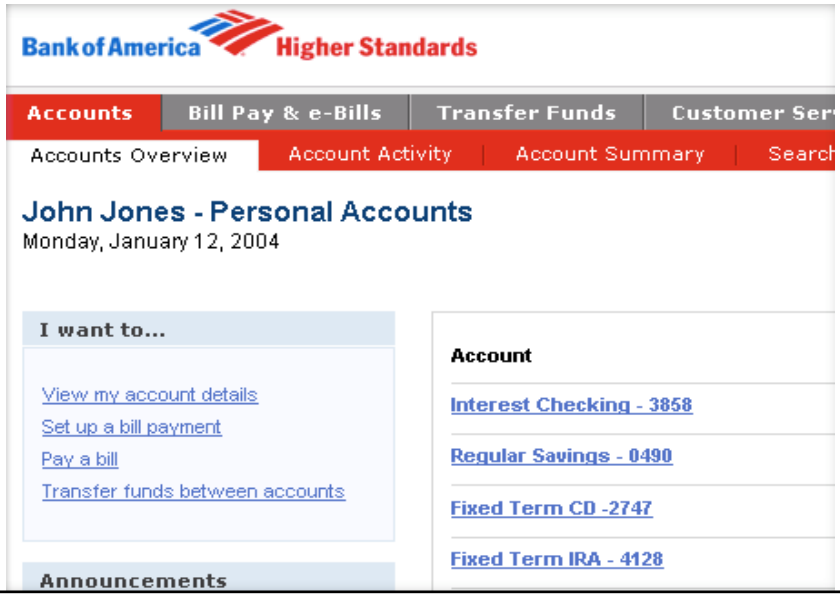
What: Database Systems Today



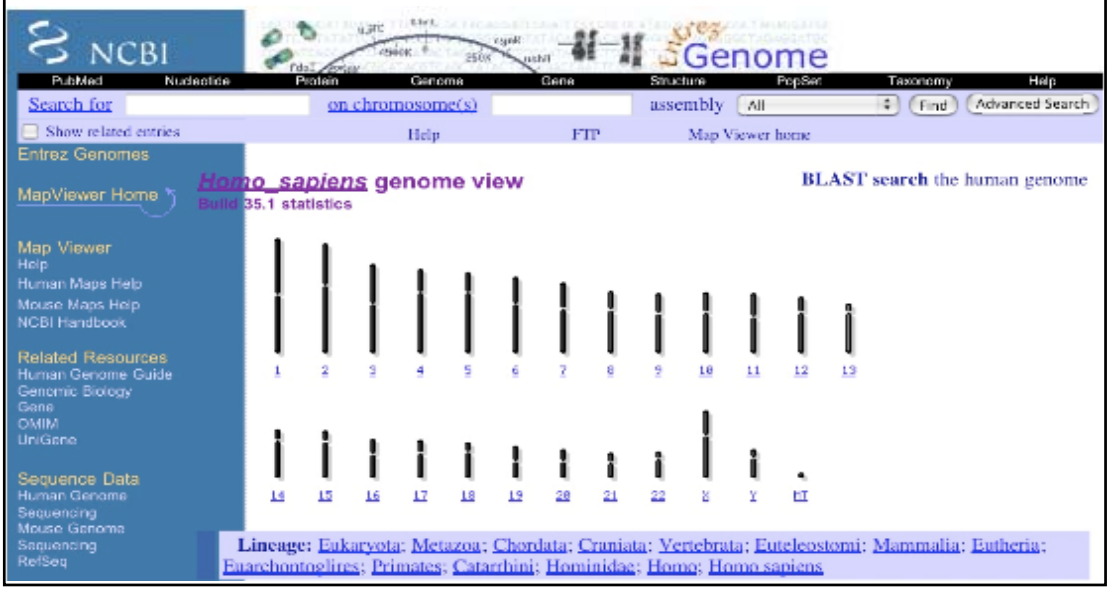
What: Database Systems Today



What: Database Systems Today



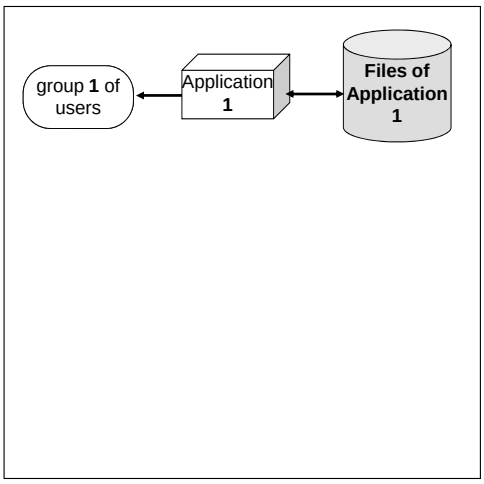
What: Database Systems Today



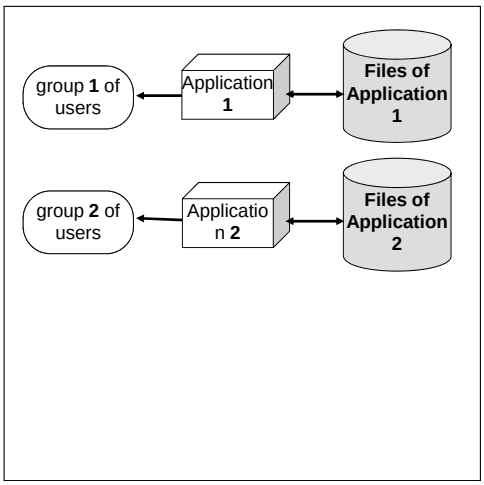
What Is a Database *System*?

- Database:
  a very large, integrated collection of data.
- Models a real-world *enterprise*
  - Entities (e.g., course, instructor)
  - Relationships
    (e.g., Tan *teaches* Database Technology)
- A *Database Management System (DBMS)* is a software system designed to store, manage, and facilitate access to databases.

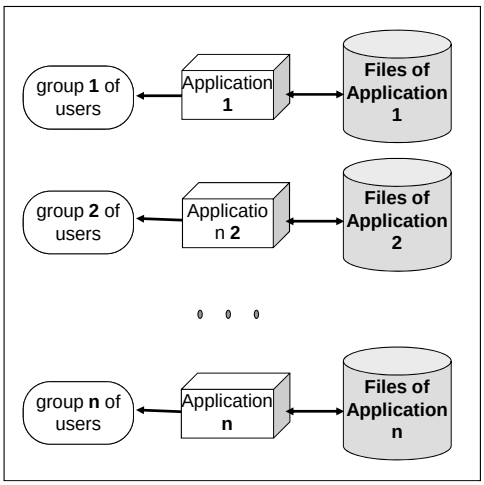
File-based vs database approach

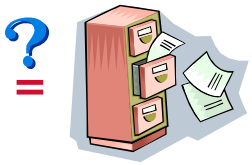


File-based vs database approach



File-based vs database approach





Is a File System a DBMS?

- Thought Experiment 1:
  - You and your project partner are editing the same file.
  - You both save it at the same time.
  - Whose changes survive?

A) Yours B) Partner's C) Both D) Neither E) ???

- Thought Experiment 2:
  - –You're updating a file.
  - –The power goes out.
  - –Which of your changes survive?

Q: How do you write programs over a subsystem when it promises you only "???" ?

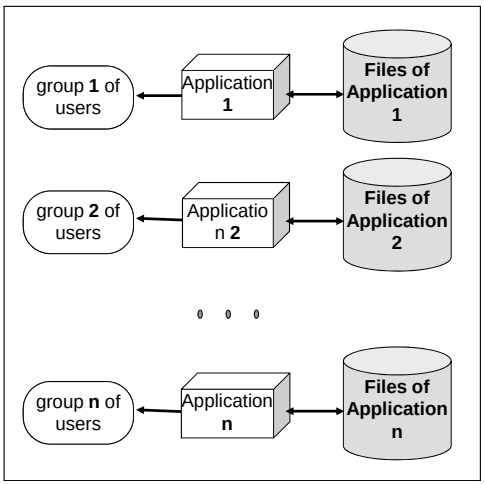
A: Very, very carefully!!

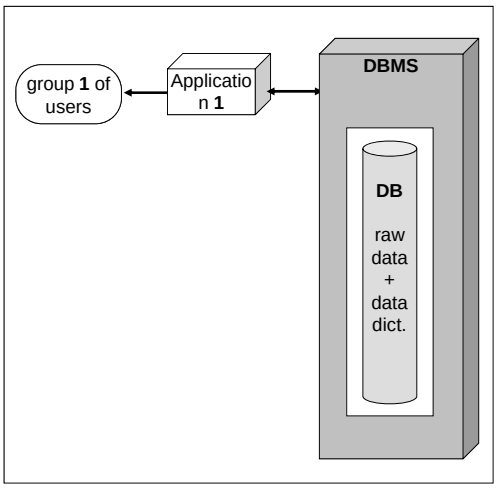
A) All B) None C) All Since last save D) ???

OS Support for Data Management

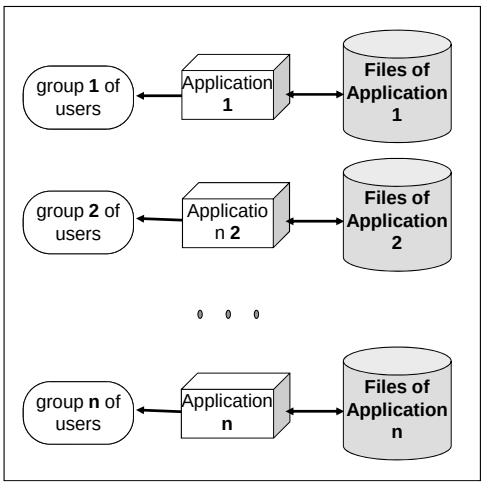
- Data can be stored in RAM
  - this is what every programming language offers!
  - RAM is fast, and random access
  - Isn't this heaven?
- Every OS includes a File System
  - manages *files* on a magnetic disk
  - allows *open, read, seek, close* on a file
  - allows protections to be set on a file
  - drawbacks relative to RAM?

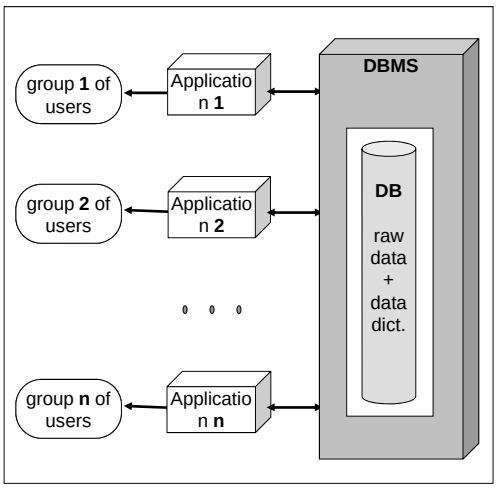
File-based vs database approach





File-based vs database approach

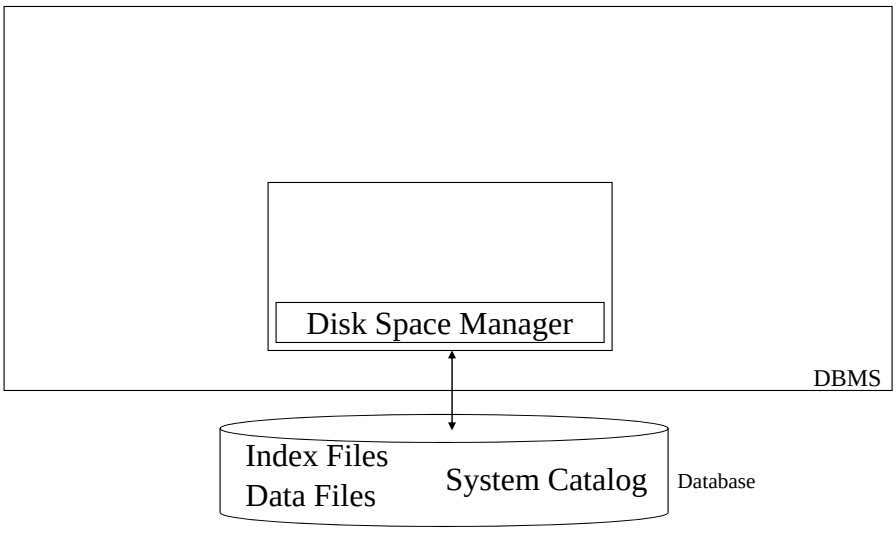




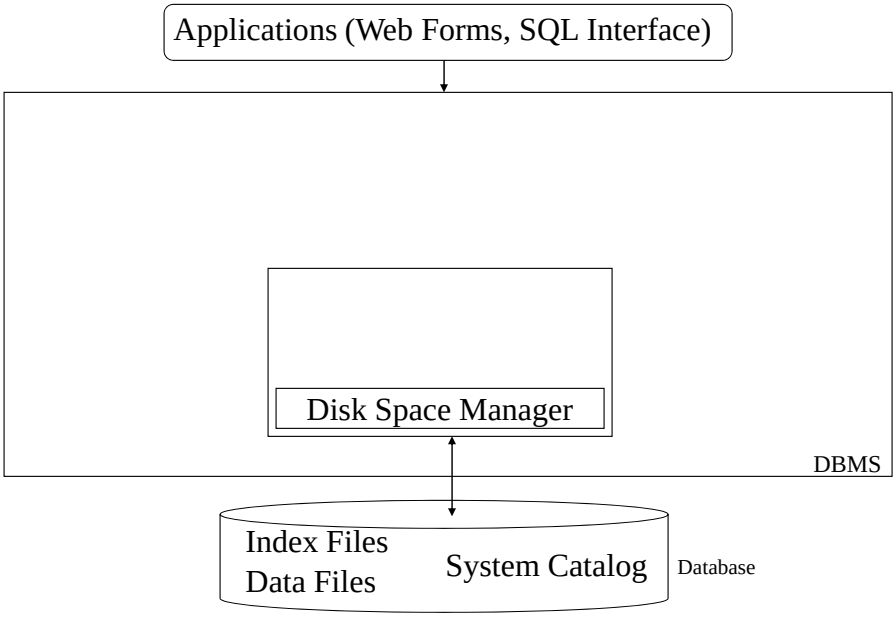
Capabilities of a Modern DBMS

- **Persistence** - permanent storage of data
- **Efficiency** - manage *large* volumes of data and *ad-hoc* queries efficiently
- **High-level access** - data model & language for defining database structures, retrieval and manipulation
- **Transaction management** - provide correct, concurrent access to the database by many users at once
- **Access control** - limit access by unauthorized users
- **Integrity management** - assure compliance to known constraints imposed by application semantics
- **Resiliency** - ability to recover from system failures without losing data

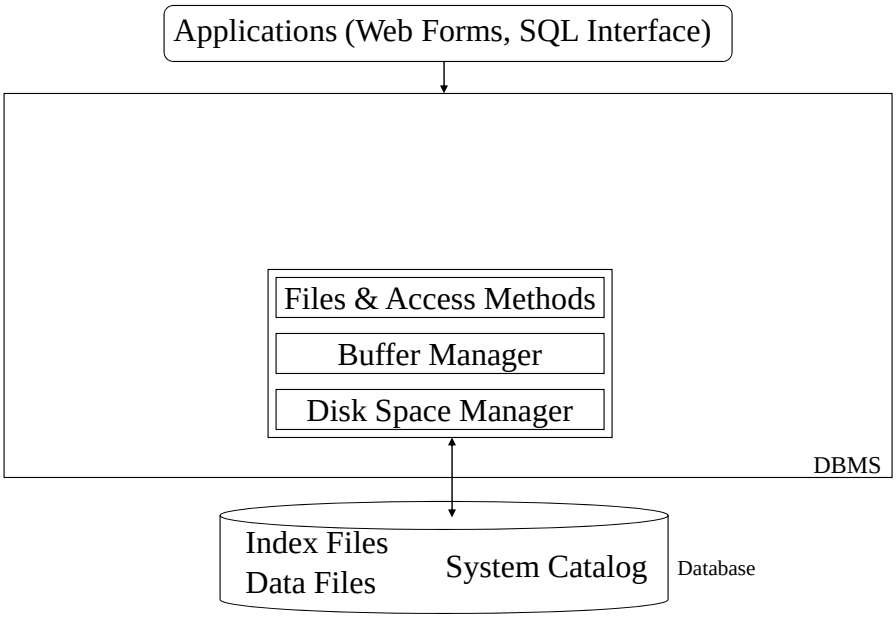
Architecture of a DBMS



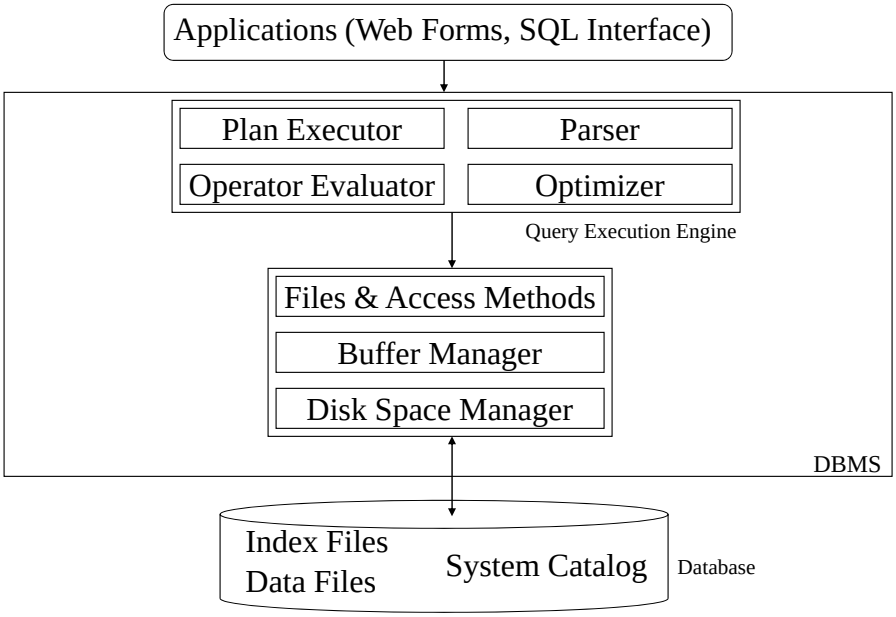
Architecture of a DBMS



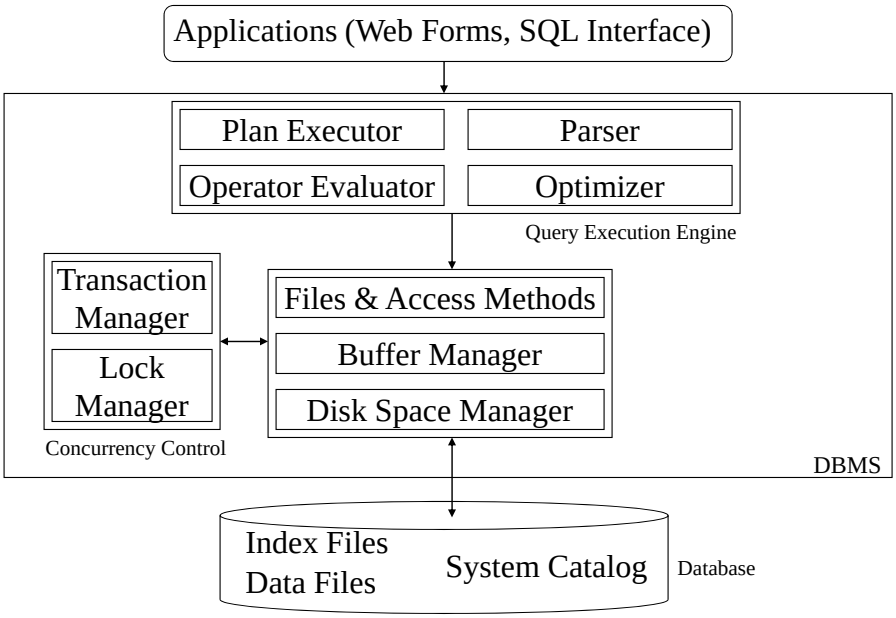
Architecture of a DBMS



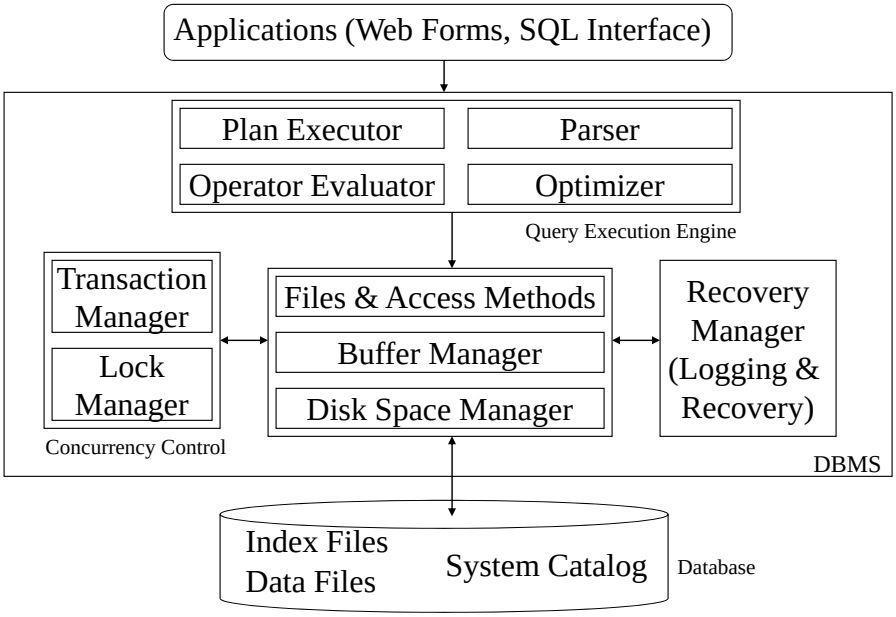
Architecture of a DBMS



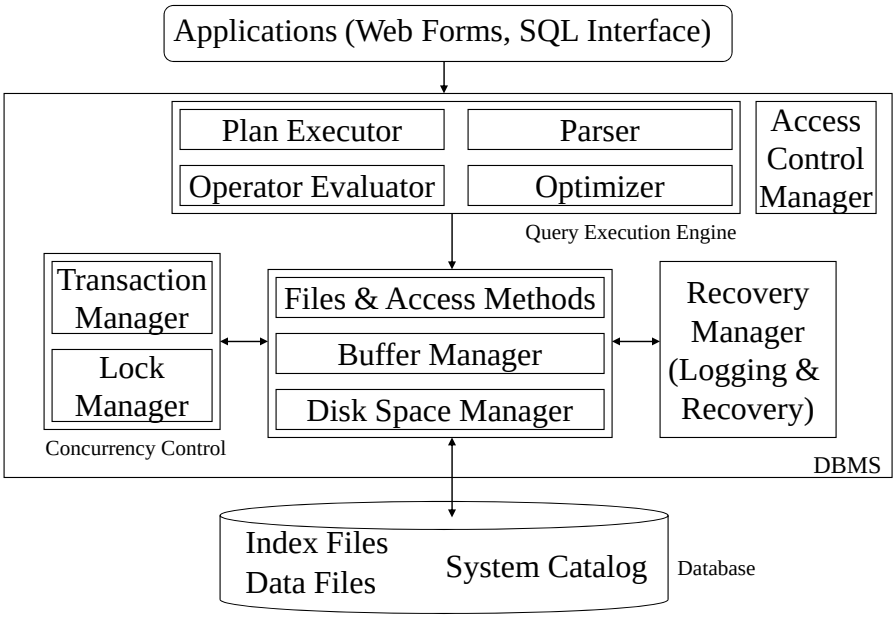
Architecture of a DBMS



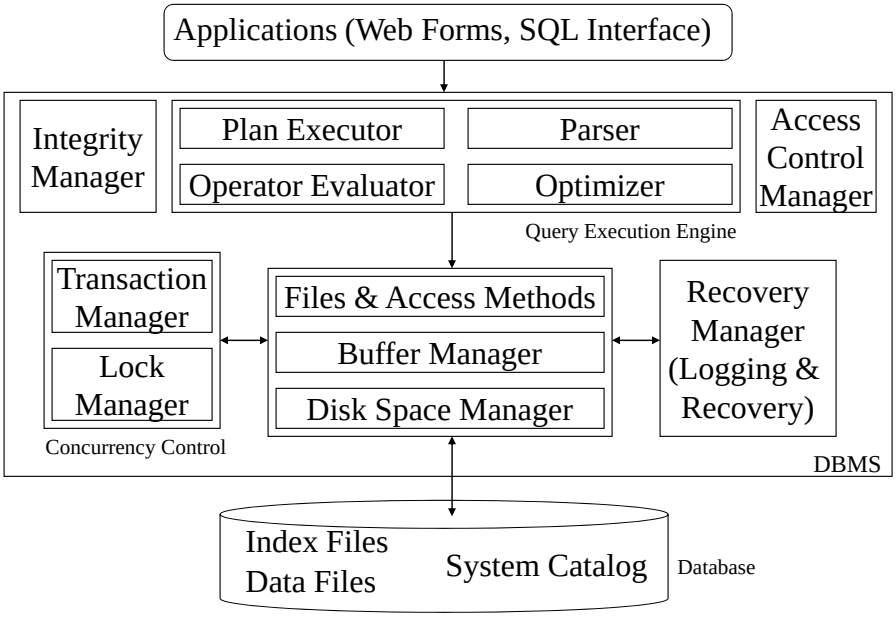
Architecture of a DBMS



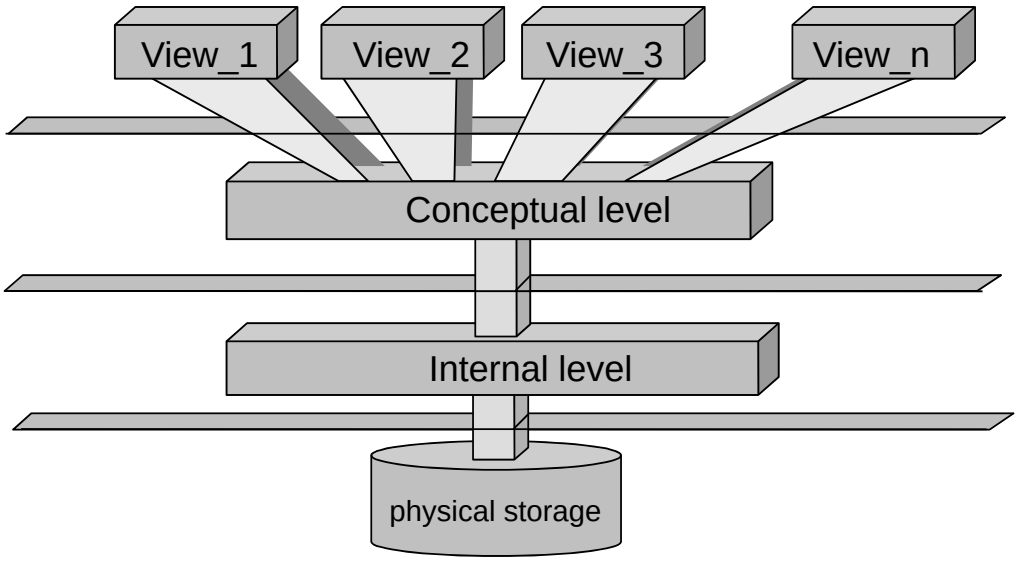
Architecture of a DBMS



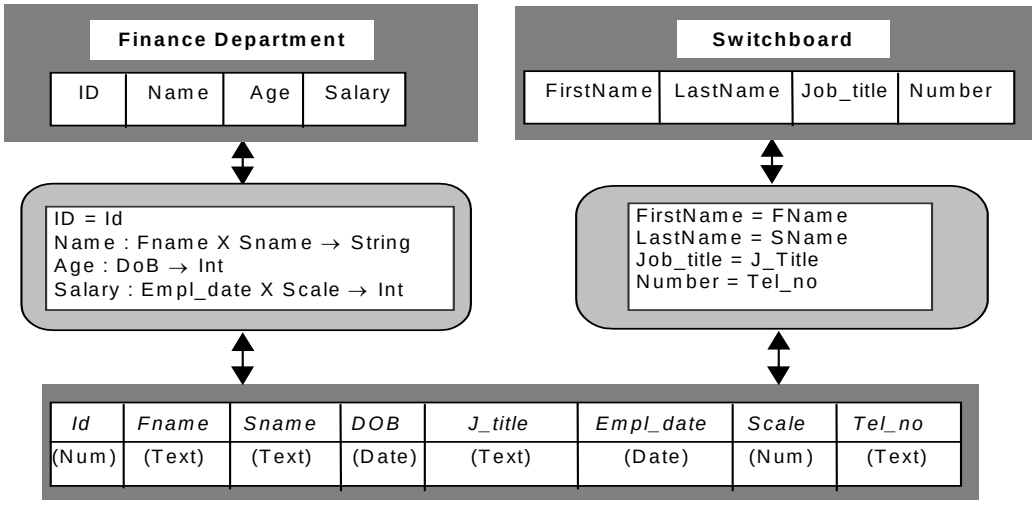
Architecture of a DBMS



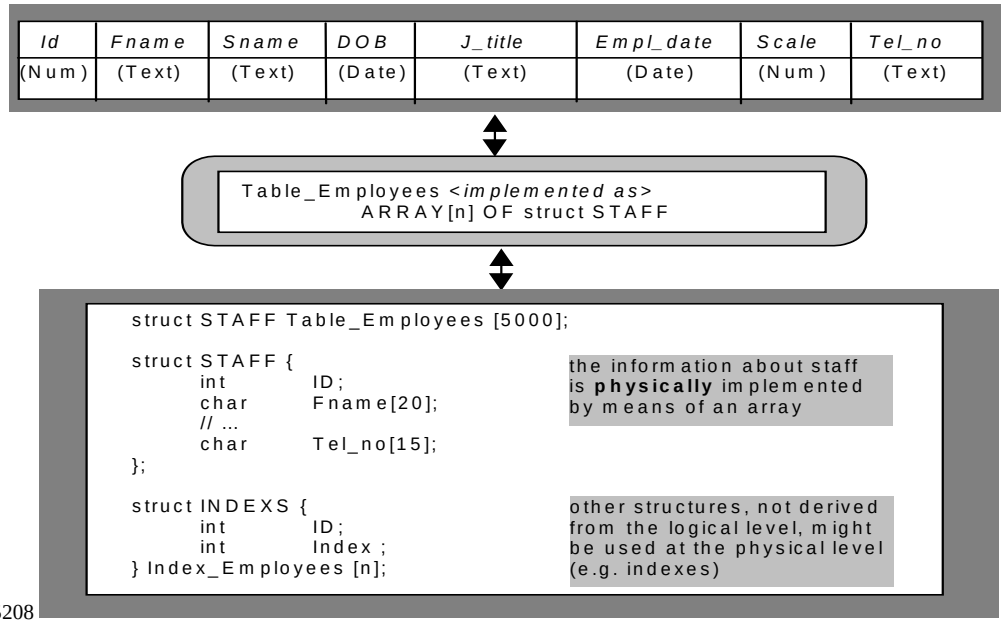
3-level Abstraction



External / conceptual example



Conceptual / internal - example



Data Independence

- Applications insulated from how data is structured and stored.
- Ability to modify a schema definition in one level without affecting a schema definition in the next higher level.
- The interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.
- *Logical* and *physical* data independence

Current Commercial Outlook

- A major part of the software industry:
  - Oracle, IBM, Microsoft
  - also Sybase, Informix (now IBM), Teradata
  - smaller players: java-based dbms, devices, OO, ...
- Well-known benchmarks (esp. TPC)
- Lots of related industries
  - data warehouse, document management, storage, backup, reporting, business intelligence, ERP, CRM, app integration
- Relational products dominant and evolving
  - adapting for extensibility (user-defined types), adding native XML support.
  - Microsoft merging file system/DB for “longhorn” (abandoned?)
- Open Source coming on strong
  - MySQL, PostgreSQL, BerkeleyDB

Why Study Databases??

- Shift from *computation* to *information*
  - "Big-Data" phenomenon
  - always true for corporate computing
  - Web made this point for personal computing
  - more and more true for scientific computing
- Need for DBMS has exploded in the last years
  - Corporate: retail swipe/clickstreams, "customer relationship mgmt", "supply chain mgmt", "data warehouses", etc.
  - Scientific: digital libraries, Human Genome project, NASA Mission to Planet Earth, physical sensors, grid physics network
- DBMS encompasses much of CS in a practical discipline
  - OS, languages, theory, AI, multimedia, logic
  - Yet traditional focus on real-world apps

Intellectual Outlook: Research Trends

- Heavy weight DBMS
  - Extend existing DBMS capabilities for advanced applications
- Light weight DBMS
  - Component-based DBMS
  - Build and use what are necessary
- Autonomic & Self tuning DBMS
  - Making the DBMS "intelligent" to minimize maintenance cost

Databases make these folks happy ...

- DBMS vendors, programmers
  - Oracle, IBM, MS, Sybase, NCR, ...
- End users in many fields
  - Business, education, science, ...
- DB application programmers
  - Build enterprise applications on top of DBMSs
  - Build web services that run off DBMSs
- Database administrators (DBAs)
  - Design logical/physical schemas
  - Handle security and authorization
  - Data availability, crash recovery
  - Database tuning as needs evolve

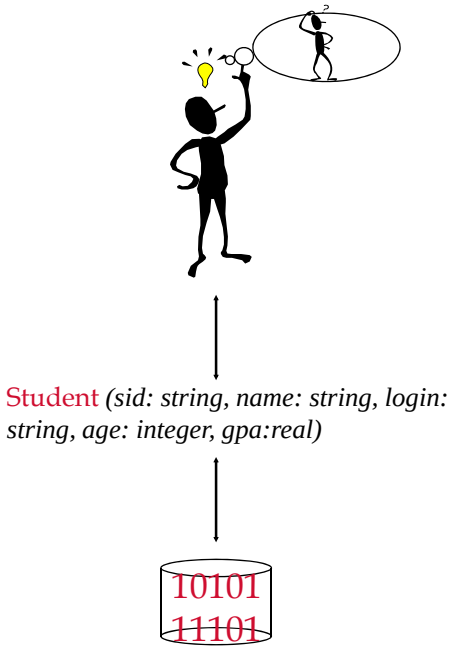
...must understand how a DBMS works

Database Design

- Why do we need it?
  - Agree on structure of the database before deciding on a particular implementation.
- Issues:
  - What to model?
  - How are things related?
  - What constraints exist?
  - How to achieve *good* designs?

Data Models

- DBMS models real world
- *Data Model* is link between user's view of the world and bits stored in computer
  - A tool for describing data, data relationships, data semantics and data constraints
- Many models exist
  - Relational Model
  - Entity-Relationship Model
  - Object-oriented Model



Entity Relationship Model

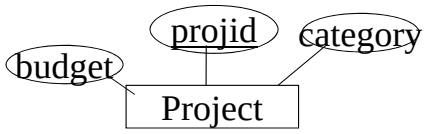
- Diagrams to represent designs

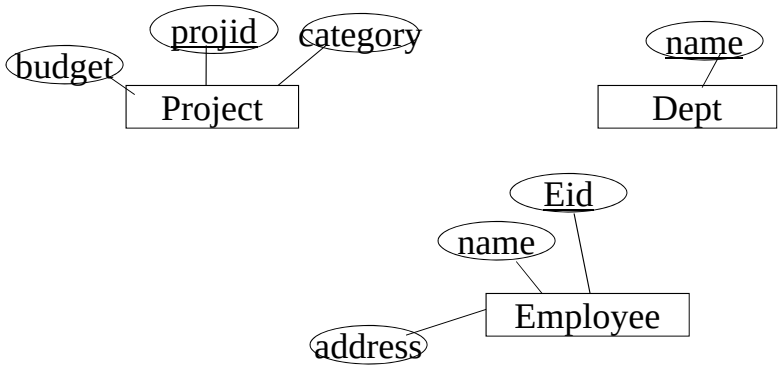
Entity = object of interest

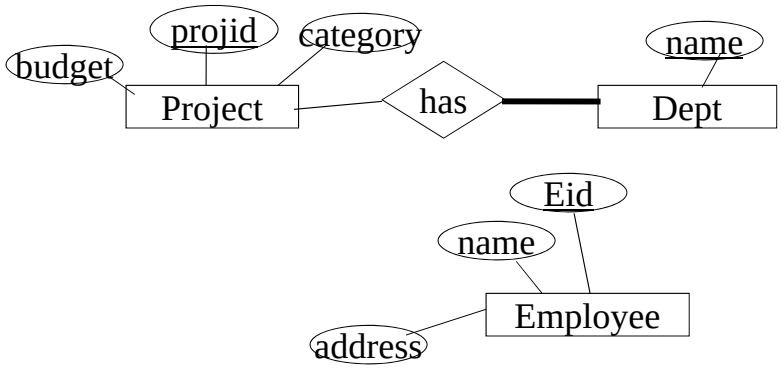
Entity set = set of similar entities.

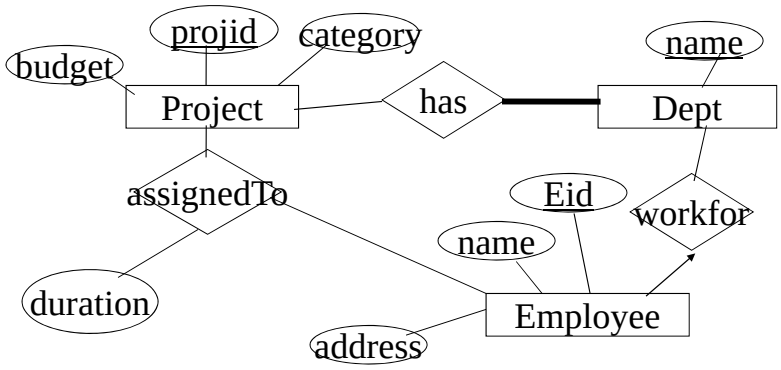
Attribute = property of entities in an entity set

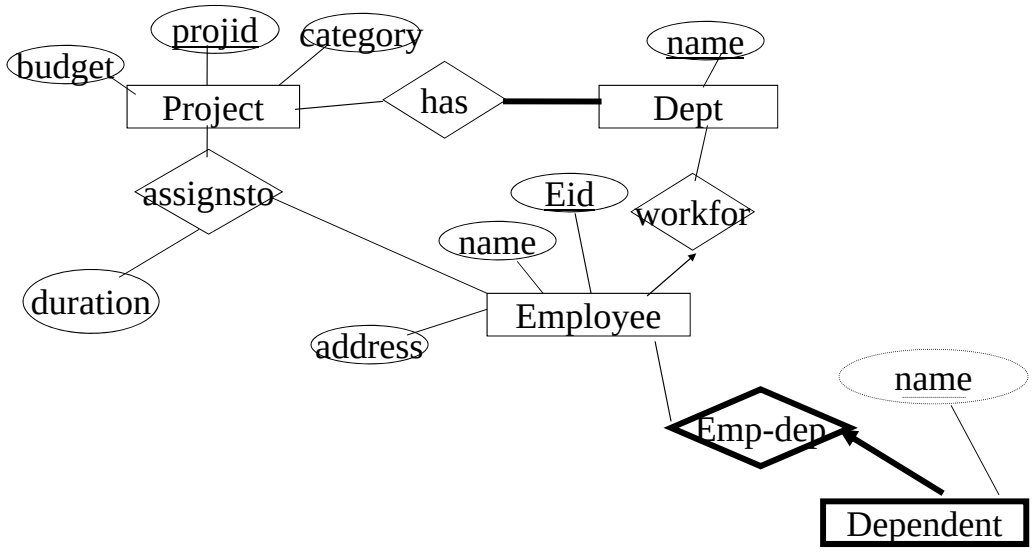
Relationship = association among entity sets

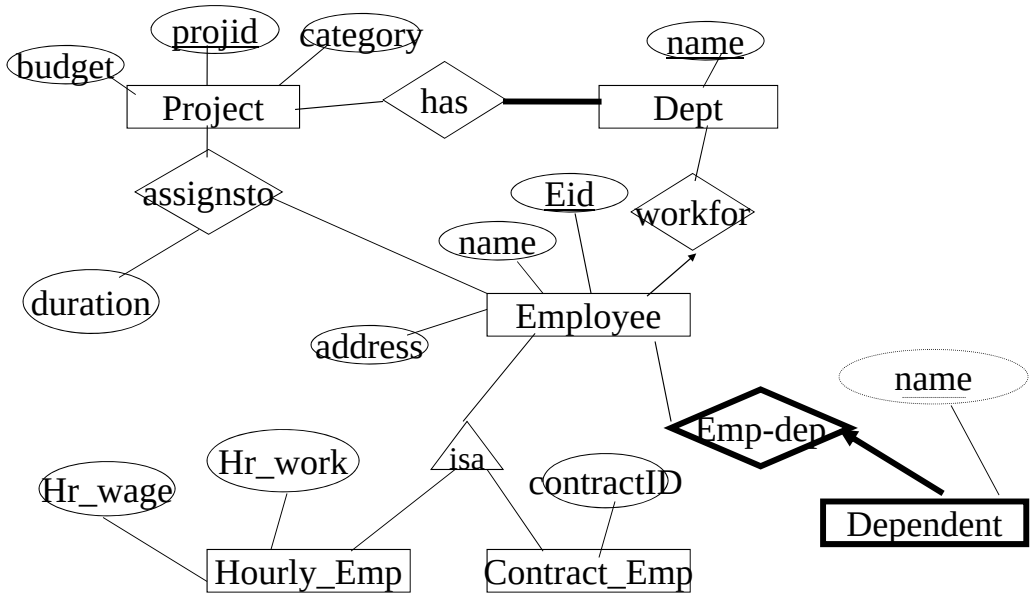

projid
category
budget
Project


projid
category
budget
Project
name
Dept
Eid
name
Employee
address


projid
category
budget
name
Project
has
Dept
Eid
name
Employee
address


projid
category
budget
name
Project
has
Dept
assignedTo
Eid
workfor
name
duration
Employee
address


projid
category
budget
Project
has
name
Dept
assignsto
Eid
workfor
name
duration
Employee
address
name
Emp-dep
Dependent


projid
category
budget
Project
has
name
Dept
assignsto
Eid
workfor
name
duration
Employee
address
name
Emp-dep
Hr_work
isa
Hr_wage
contractID
Dependent
Hourly_Emp
Contract_Emp

Relational Data Model

- a data model in which all data is modelled as relations (tables)
  - a way of looking at data
- a prescription for a way of
  - representing data
  - manipulating data (relational algebra)
  - representing integrity constraints

A Relation

A relation contains a SET of tuples

PARTS(Name: String; Price: Real; Category: String; Manufacturer: String)

Attribute names

Name	Price ($)	Category	Manufacturer
Gizmo	19.99	gadgets	GizmoWorks
Power gizmo	29.99	gadgets	GizmoWorks
SingleTouch	149.99	photography	Canon
MultiTouch	203.99	household	Hitachi

Tuples (record)

Each attribute has an atomic type

Integrity

- restrictions on data defined by users
  - on individual tables
    - age > 18; salary < 100k
  - on more than one table
    - *if* budget < 10M *then* salary < 50k
- implicit in the data model

Integrity Constraints (ICs)

- IC: condition that must be true for *any* instance of the database
  - e.g., *domain constraints*
    - Each attribute has values taken from a *domain*. ICs are specified when schema is defined.

Primary and foreign keys

E_id	E_name	Dept_id	Salary
E1	Smith	D1	40K
E2	John	D1	42K
E3	Stella	D2	30K
E4	Art	D3	35K

foreign

primary

Dept_id	Dept_name	Budget
D1	Marketing	10M
D2	Development	12M
D3	Research	5M

More Integrity Constraints

- *Key constraints*: each tuple must be distinct. A key is a subset of fields that uniquely identifies a tuple (*superkey*), and for which no subset of the key has this property.
- *Referential integrity constraints*: a field in one relation may refer to a tuple in another relation by including its key (*foreign key*). The referenced tuple must exist in the other relation for the database instance to be valid.
- Typically, a relation may have several *candidate* keys one of which is chosen as the *primary* key.

From E/R Diagrams to Relational Schema

Database Model (E/R) → Relational Schema → Physical storage

Diagrams (E/R)

Tables:
columns: attributes
rows: tuples

Complex file organization and index structures.

From E/R Diagrams to Relational Schema

Database Model (E/R) → Relational Schema → Physical storage

Diagrams (E/R)

Tables:
columns: attributes
rows: tuples

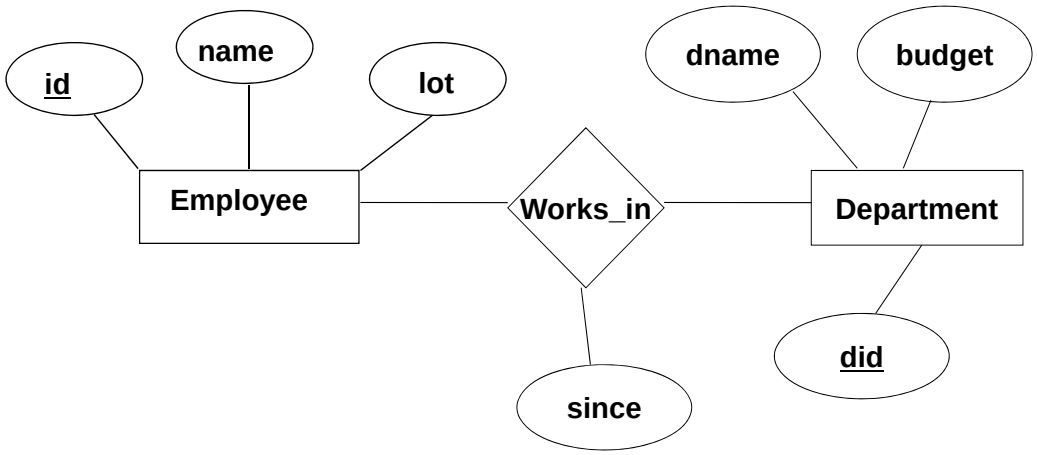
Complex file organization and index structures.

Entity sets → relations
Relationships → relations
Special care for weak entity sets.

Relationships to Relations

- In translating a relationship set to a relation, attributes of the relation must include:
  - Keys for each participating entity set (as foreign keys).
    - This set of attributes forms a ***superkey*** for the relation.
  - All descriptive attributes.

Example



Example continued

Relation Employee
id : CHAR(9),
name : CHAR(20),
lot : INTEGER
PRIMARY KEY id

Example continued

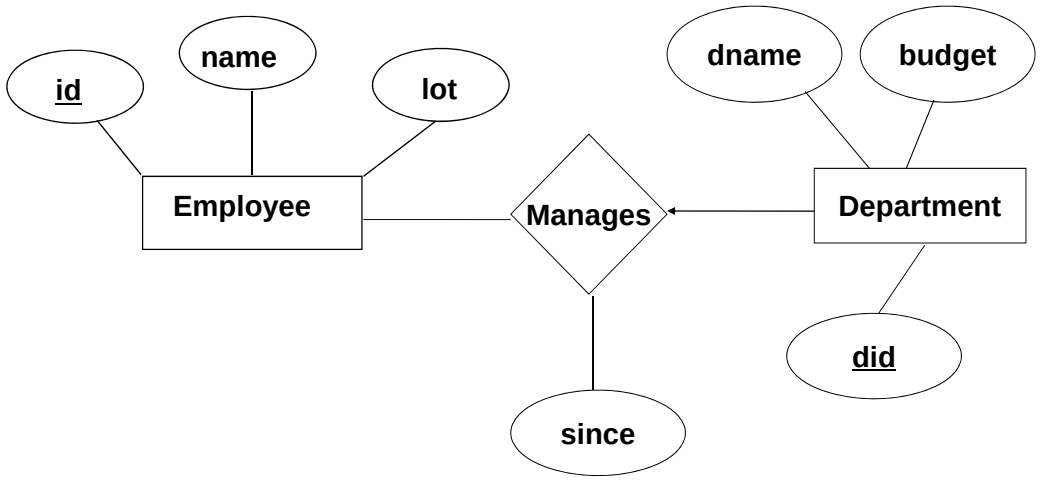
Relation Employee
id : CHAR(9),
name : CHAR(20),
lot : INTEGER
PRIMARY KEY id

Relation Department
did : INTEGER,
dname : CHAR(20),
budget : REAL
PRIMARY KEY did

Relation Works_In
id : CHAR(9),
did : INTEGER,
since : DATE,
PRIMARY KEY (id, did),
FOREIGN KEY (id) REFERENCES Employee,
FOREIGN KEY (did) REFERENCES Department

Key constraints

Each dept has at most one manager, *key constraint* on Manages.



Key Constraints

- Map relationship to a table:
  - Note that ***did*** is the key now!

```
Relation  Manages
 id  : CHAR(9),
 did : INTEGER,
 since :  DATE,
 PRIMARY KEY  did,
 FOREIGN KEY  id  REFERENCES Employee,
 FOREIGN KEY did  REFERENCES Department
```

Key Constraints

Since each department has a unique manager, we could instead combine Manages and Departments.

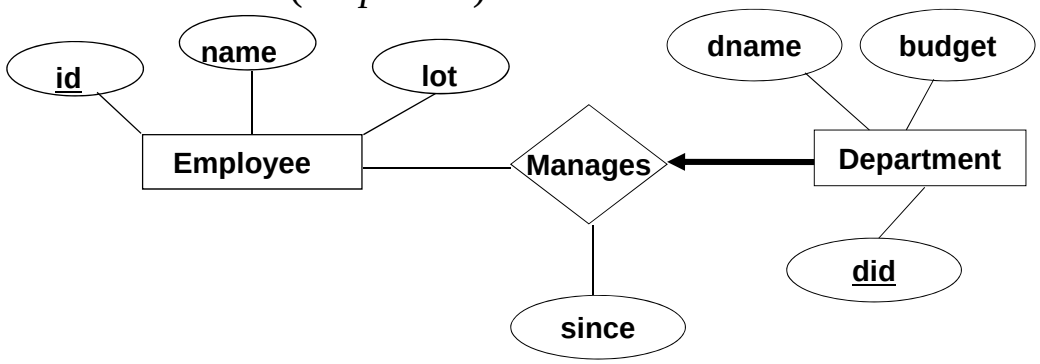
Key Constraints

Since each department has a unique manager, we could instead combine Manages and Departments.

```
Relation   Dept_Mgr
   did  : INTEGER,
   dname  : CHAR(20),
   budget  : REAL,
   id :  CHAR(11),
   since  : DATE,
   PRIMARY KEY  did,
   FOREIGN KEY id REFERENCES Employee
```

Participation Constraints

- Does every department have a manager?
  - If so, this is a *participation constraint*: the participation of Departments in Manages is said to be *total* (vs. *partial*).



Participation Constraints

- Every *did* value in Department table must appear in a row of the Manages table (with a non-null id value!)

Participation Constraints

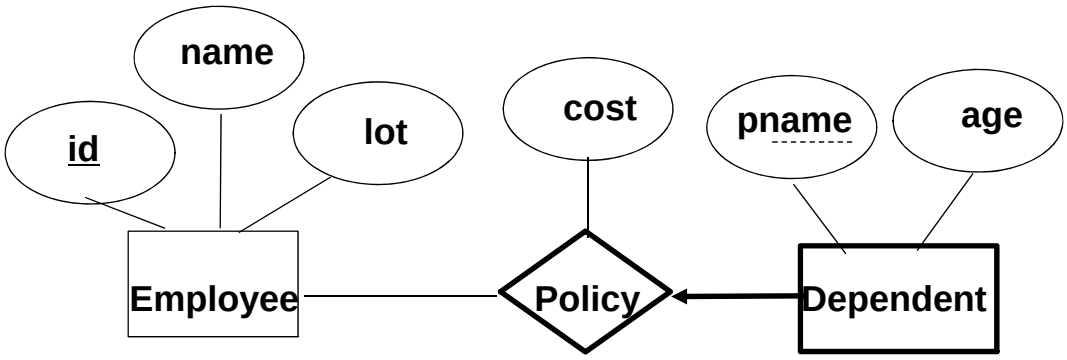
- Every *did* value in Department table must appear in a row of the Manages table (with a non-null id value!)

```
Relation  Dept_Mgr
  did  : INTEGER,
  dname  : CHAR(20),
  budget  : REAL,
  id  : CHAR(9) NOT NULL,
  since : DATE,
  PRIMARY KEY  did,
  FOREIGN KEY  (id) REFERENCES Employee,
  ON DELETE NO ACTION
```


Weak Entities to Relations

- A *weak entity* can be identified uniquely only by considering the primary key of another (*owner*) entity.
  - a one-to-many relationship set (1 owner, many weak entities).
  - Weak entity set must have total participation in this *identifying* relationship set.

Weak Entities to Relations



Translating Weak Entity Sets

- Weak entity set and identifying relationship set are translated into a single table.
  - When the owner entity is deleted, all owned weak entities must also be deleted.

```
Relation  Dep_Policy
  pname  CHAR(20),
  age  INTEGER,
  cost  REAL,
  id  CHAR(9) NOT NULL,
  PRIMARY KEY  (pname, id),
  FOREIGN KEY  (id) REFERENCES Employee,
    ON DELETE CASCADE
```

Translating ISA Hierarchies

- 3 relations: Employees, Hourly_Emps and Contract_Emps.
- *Hourly_Emps*: Every employee is recorded in Employees.
  - extra info recorded in Hourly_Emps (*hourly_wages*, *hours_worked*, id);
  - must delete Hourly_Emps tuple if referenced Employees tuple is deleted.

You are now a *trained* database designer!