Basic Concepts of Database Systems

Example Uses of Database Systems

- account maintenance & access in banking
- lending library systems
- airline reservation systems
- internet purchasing systems
- media archives for radio/tv stations

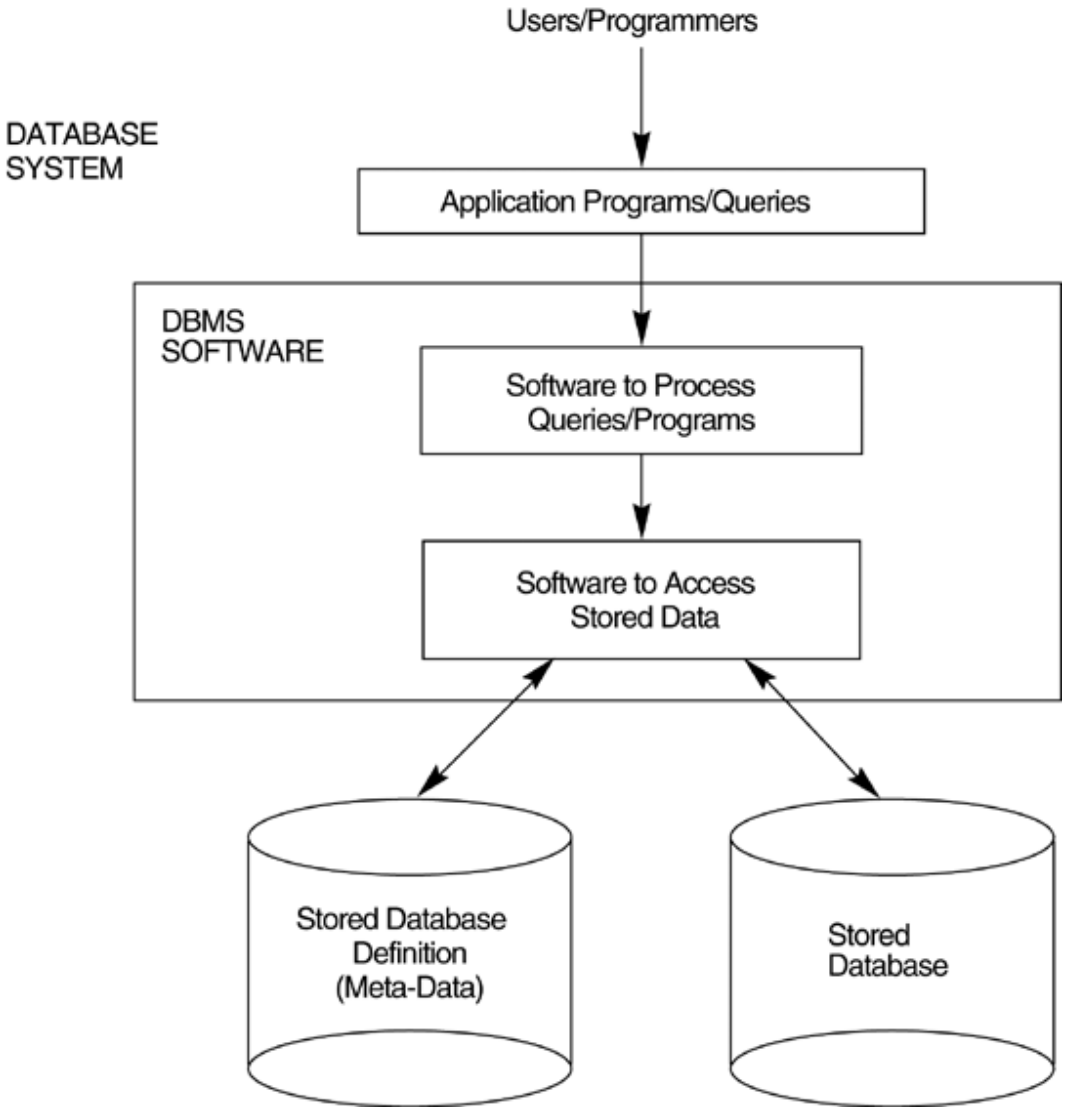
Definition of Terms

- ***Database:***
    - A logically coherent collection of related data that (i) describes the entities and their inter-relationships, and (ii) is designed, built & populated for a specific reason

- ***Database Management System (DBMS):***
    - A collection of programs that enables users to perform certain actions on a particular database:
        - *define* the structure of database information (descriptive attributes, data types, constraints, etc), storing this as meta-data
        - *populate* the database with appropriate information
        - *manipulate* the database (for retrieval/update/removal/insertion of information)
        - *protect* the database contents against accidental or deliberate corruption of contents (involves secure access by users and automatic recovery in the case of user/hardware faults)
        - *share* the database among multiple users, possibly concurrently

[Example DBMS would include Oracle, Sybase, MySQL, DB/2, SQLServer, Informix, MS-Access, FileMaker, …]

- ***Database System:***
    - Database System = Database + Database Management System



Sample Databases

Shown below is an extract from a (relational) database that might be part of a University's Academic Information System:

STUDENT	Name	StudentNumber	Class	Major
	Smith	17	1	CS
	Brown	8	2	CS

COURSE	CourseName	CourseNumber	CreditHours	Department
	Intro to Computer Science	CS1310	4	CS
	Data Structures	CS3320	4	CS
	Discrete Mathematics	MATH2410	3	MATH
	Database	CS3380	3	CS

SECTION	SectionIdentifier	CourseNumber	Semester	Year	Instructor
	85	MATH2410	Fall	98	King
	92	CS1310	Fall	98	Anderson
	102	CS3320	Spring	99	Knuth
	112	MATH2410	Fall	99	Chang
	119	CS1310	Fall	99	Anderson
	135	CS3380	Fall	99	Stone

GRADE_REPORT	StudentNumber	SectionIdentifier	Grade
	17	112	B
	17	119	C
	8	85	A
	8	92	A
	8	102	B
	8	135	A

PREREQUISITE	CourseNumber	PrerequisiteNumber
	CS3380	CS3320
	CS3380	MATH2410
	CS3320	CS1310

Shown below is an extract from a (relational) database that might be part of a Consultancy Business Information System:

EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

DEPT_LOCATIONS	DNUMBER	DLOCATION
	1	Houston
	4	Stafford
	5	Bellaire
	5	Sugarland
	5	Houston

DEPARTMENT	DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
	Research	5	333445555	1988-05-22
	Administration	4	987654321	1995-01-01
	Headquarters	1	888665555	1981-06-19

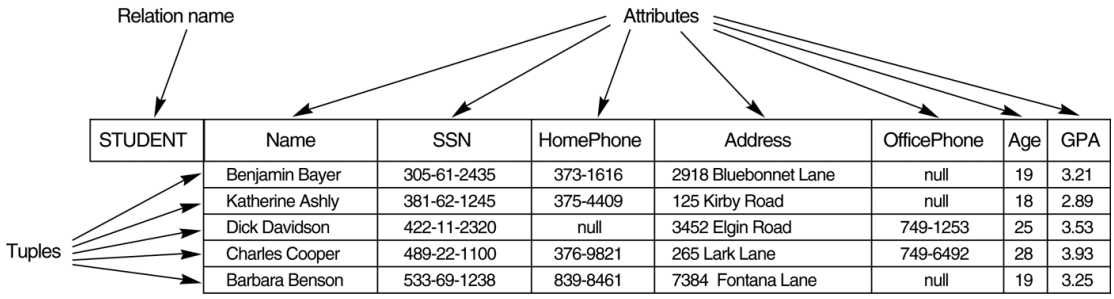
WORKS_ON	ESSN	PNO	HOURS
	123456789	1	32.5
	123456789	2	7.5
	666884444	3	40.0
	453453453	1	20.0
	453453453	2	20.0
	333445555	2	10.0
	333445555	3	10.0
	333445555	10	10.0
	333445555	20	10.0
	999887777	30	30.0
	999887777	10	10.0
	987987987	10	35.0
	987987987	30	5.0
	987654321	30	20.0
	987654321	20	15.0
	888665555	20	null

PROJECT	PNAME	PNUMBER	PLOCATION	DNUM
	ProductX	1	Bellaire	5
	ProductY	2	Sugarland	5
	ProductZ	3	Houston	5
	Computerization	10	Stafford	4
	Reorganization	20	Houston	1
	Newbenefits	30	Stafford	4

DEPENDENT	ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
	333445555	Alice	F	1986-04-05	DAUGHTER
	333445555	Theodore	M	1983-10-25	SON
	333445555	Joy	F	1958-05-03	SPOUSE
	987654321	Abner	M	1942-02-28	SPOUSE
	123456789	Michael	M	1988-01-04	SON
	123456789	Alice	F	1988-12-30	DAUGHTER
	123456789	Elizabeth	F	1967-05-05	SPOUSE

Terminology:

- relation = table (file)
- attribute = column (field)
- tuple = row (record)



STUDENT	Name	SSN	HomePhone	Address	OfficePhone	Age	GPA
	Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	null	19	3.21
	Katherine Ashly	381-62-1245	375-4409	125 Kirby Road	null	18	2.89
	Dick Davidson	422-11-2320	null	3452 Elgin Road	749-1253	25	3.53
	Charles Cooper	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
	Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	null	19	3.25

Concepts:

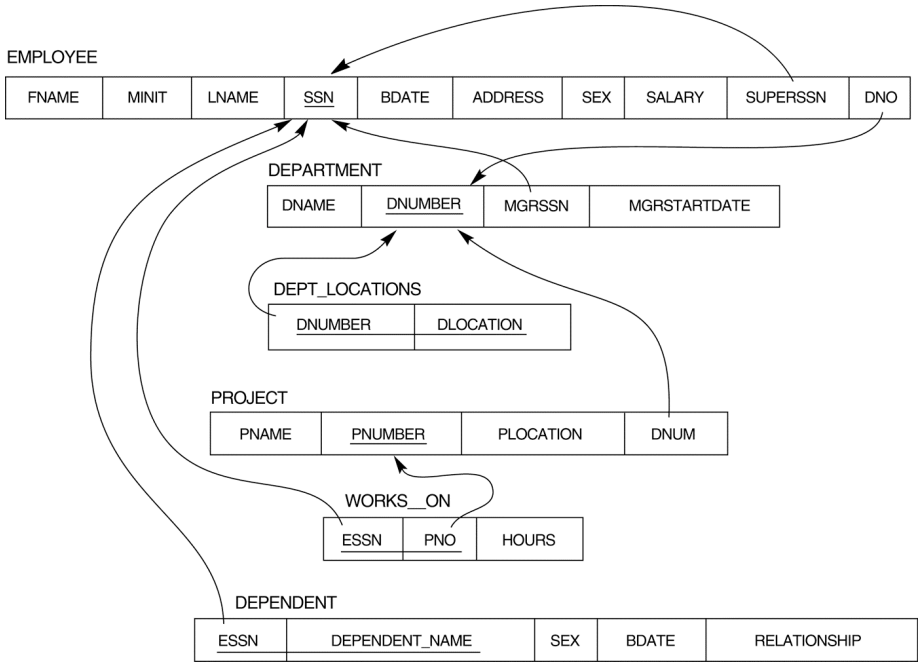
- Domains:
  - an attribute will always be chosen from some *domain* of valid values (e.g. SSN will be chosen from the valid 9-digit collection of Social Security Nos., GPA will be chosen from the 1.2-digit set of values)
- Null Values:
  - some attributes may be allowed to assume *null* (unspecified) values – typically where an attribute may be inapplicable or just unknown at the point of data entry
- Row Ordering:
  - the order of tuples within a relation is unimportant [in practice, it may be, since it can determine the order of returned data rows
- Column Ordering:
  - the order of attributes within a relation is unimportant [in practice, it may be, since it can determine the order of returned data columns]
- Key:
  - every relation should have an attribute (possibly composite) that behaves as a *key* – it is unique in value to each tuple, and can

therefore act as an identifier for a particular tuple; e.g., in the STUDENT table immediately above, SSN acts as the key attribute; in the GRADE_REPORT relation seen earlier, the key is the *composite attribute* (StudentNumber, SectionIdentifier)

- note that combining the key with any collection of other attributes also guarantees uniqueness – e.g., in the STUDENT table, the combination (SSN, Name), or any other combination involving SSN, can act as identifier; we call these *superkeys* and define a *key* as the minimal collection of attributes that guarantees uniqueness (the *minimal superkey*)
- it may sometimes occur that we have more than one choice of key attribute within a relation – e.g., in the COMPANY database seen at the start of this lecture, we may have knowledge that both department numbers (DNUMBER) and names (DNAME) are unique; we declare these to be *candidate keys* and generally choose one to be the *primary key* (DNUMBER, say) and the others to be *alternate keys*

- Key Integrity (Entity Constraints):
  - no component of a candidate key should be allowed to accept null values; no replicate key values should be allowed

- Foreign Key:
  - if a relation R1(…, X, …) contains an attribute X that acts as a key to another relation R2(…, X, …), then we say that R1 *references* R2; attribute X is said to be a *foreign key* within R1; we can depict all such references & foreign keys within a database by incorporating them into the schema, as outlined below for the COMPANY database:



- Referential Integrity Constraints:
    - if a relation R1(…, X, …) references another relation R2(…, X, …) on attribute X, then every value of X appearing in R1 must also appear in R2

Database Users

- Database Administrators (DBA):
    - individual(s) that determine & implement policy regarding users, their permissions on a database and the design & construction of that database
- Database Designers:
    - individual(s) – possibly also *software engineers* – who apply design techniques to produce database structures pertinent to a specific application
- End Users:
    - People who, from time to time, access the contents of a database:
        - *Casual end users* may submit ad-hoc queries as the need arises, using a high-level query language
        - *naïve,* or *parametric*, *end-users* access the database through pre-written programs that effect an appropriate interface to the database
        - *database programmers* write code, using a relevant programming language and the high-level query language, that can later be used by parametric users

Database Languages & Interfaces

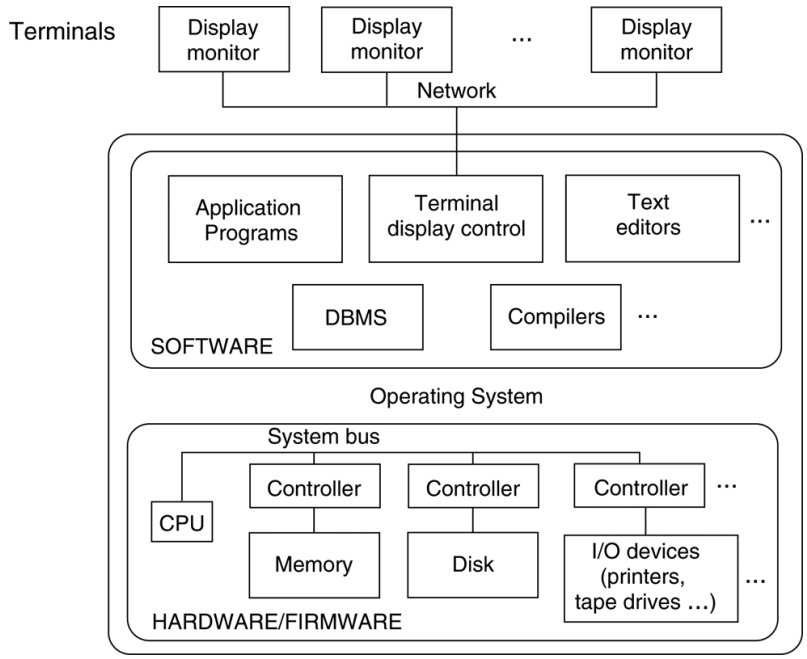
- since the inception of the relational data model, DBMSs have supported a set of instructions for user interaction; the instructions constitute a *data sublanguage (DSL)*
- any DSL can be divided into three instruction sets:
    - the *data definition language (DDL)* comprises those instructions used for creating, removing and altering data structures for containing information
    - the *data manipulation language (DML)* comprises those instructions used for retrieval, deletion, update & insertion of database contents

  - the *data control language (DCL)* comprises those instructions used for specifying access permissions on the database structures & contents

- *Structured Query Language (SQL)*, originally specified as part of the System R project, is a DSL that is common to many implementations of the relational model: Oracle, Sybase, SQLServer, MySQL, DB/2, …
- beyond this command-oriented interaction, many users may benefit from having more user-friendly interfaces – partly or wholly graphical in nature; such facilities might be provided as part of DBMS, might be coded externally by programmers or might be supplied by third-party add-on providers; interface types might include:
  - Menu-based interfaces for Web clients or browsing
  - Forms-based interfaces
  - Graphical user interfaces
  - Natural language interfaces
  - Interfaces for parametric users
  - Interfaces for the DBA

DBMS Execution Environment

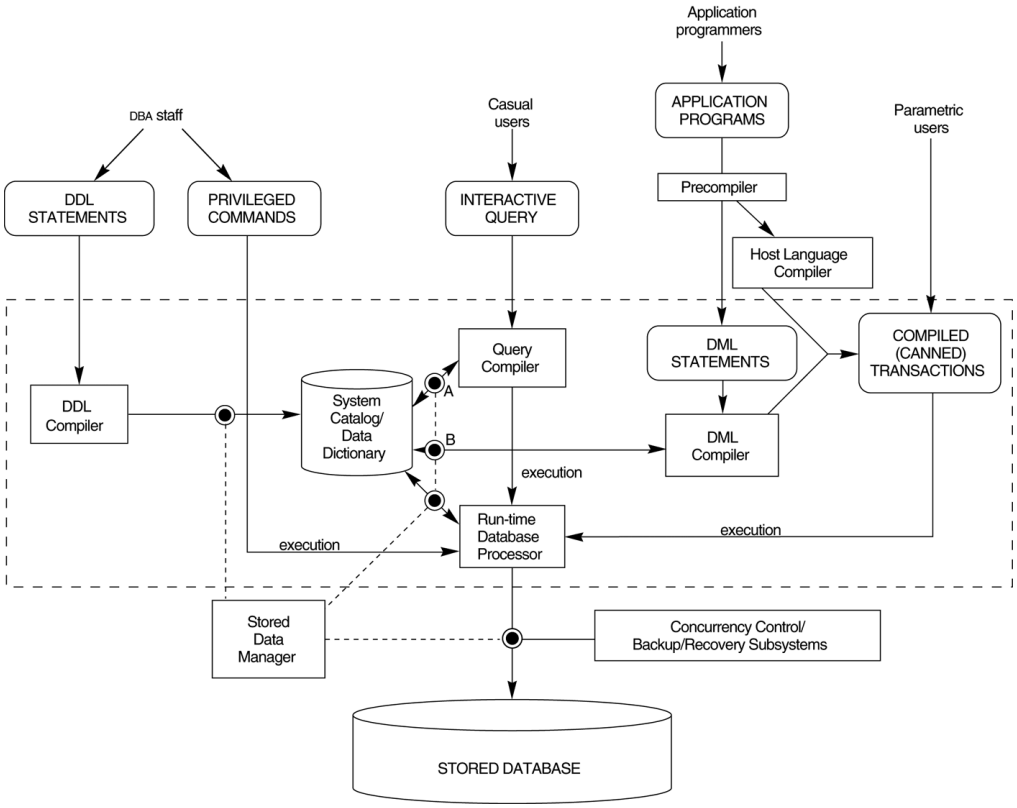
- a DBMS behaves just like any application program: it executes as a process on top of the resident operating system (albeit, perhaps, at a high priority):



- the DBMS uses the OS features as does any other application: for memory & CPU allocation, for I/O to/from display & keyboard … and particularly for storage of data on disk

Database Architecture

- the collection of programs that comprise a DBMS can be organized into a set of component modules, each of which carry out a specific task:



[cornered rectangles represent DBMS component modules; rounded rectangles represent user commands or programs; all activities and communications within dotted rectangle are under control of Stored Data Manager]

- these necessary components might be augmented by *utility programs* that provide useful services:
  - for *loading* of data from data files or other storage formats
  - for *backup* of database contents – totally or incrementally – onto a save storage medium (tape, say)
  - for *reorganization* of stored data to enhance space/time efficiency

  - for *performance monitoring* of DBMS by DBA so as to optimize response time, throughput, etc.

- also, for application program development, internal/external software systems might be provided with connectivity to the DBMS (*application development environments)* – such as Jbuilder from Borland or PowerBuilder from Sybase
- finally, many business enterprises may have several disjoint databases (and even DBMS), may have applications using files only and may have purely manual processing of some data; in order to keep track of all data used (data item names, aliases, data types, inter-relationships, etc), a software *data dictionary system* might be employed; it is useful if there are automatic import/export facilities between the database system catalog and the wider data dictionary