Course Notes on
Databases and Database Management Systems

Databases and Database Management Systems: Summary

- Databases
- Database management systems
- Schema and instances
- General view of DBMS architecture
- Various levels of schema
- Integrity constraint management
- Notion of data model
- Database languages and interfaces
- Other DBMS functions
- Roles and functions in database management

1

Database

- A collection of related data with
  - logically coherent **structure**
  - inherent **meaning**
  - **purpose**, for intended users and applications
  - varying **size**
  - scope, **content** of varying breadth
  - **physical organization** of varying complexity
  - various applications with **possibly-conflicting objectives**
  - **persistence**, existence over a long period of time

2

- **Database** = a collection of related data with
  - a logically coherent **structure** (can be characterized as a whole)
  - some inherent **meaning** (represents some partial view of a portion of the real world)
  - a specific **purpose**, an intended group of users and applications (a database embodies a biased, **operational view on the world**; database management is not after modeling the world in general, maybe philosophy or ontology are)
  - a largely **varying size** (from a personal list of addresses to the National Register of Persons)
  - a scope or **content** of varying breadth (from a personal list of addresses to a multimedia encyclopedia)
  - a **physical organization of varying complexity** (from a manual personal list, managed with simple files, to huge multi-user databases with geographically distributed data and access)
  - logically-coordinated **objectives**, data is defined once for a community of users, and accessed by **various applications** with specific needs

Database Management Systems (DBMSs)

- DBMS: a collection of general-purpose, **application-independent programs** providing services to
  - **define the structure** of a database, i.e., data types and constraints that the data will have to satisfy
  - **manage the storage** of data, safely for long periods of time, on some storage medium controled by the DBMS
  - **manipulate a database**, with efficient user interfaces to query the database to retrieve specific data, update the database to reflect changes in the world, generate reports from the data
  - **manage database usage**: users with their access rights, performance optimization, sharing of data among several users, security from accidents or unauthorized use
  - **monitor** and analyze database usage

3

- DBMS have similarities with operating systems: both manage memory, process scheduling, I/O, communication
- In addition, DBMSs implement many data-management functions
- Other name for DBMS: database system, database manager
- DBMSs typically do not use the file system of the operating system of the machine where they are installed. Instead, the define their own richer file organizations and access methods

Example of a Database

Student

StudName	StudNo	Class	Dept
Smith	17	1	CS
Brown	8	2	CS

Course

CourseName	CourseNo	Credits	Dept
Introduction to CS	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MA2410	3	MA
Database Management	CS3380	3	CS

Prerequisite

CourseNo	PrereqNo
CS3380	CS3320
CS3380	MA2410
CS3320	CS1310

Session

SessIdent	CourseNo	Semester	Year	Professor
85	MA2410	Fall	96	King
92	CS1310	Fall	96	Anderson
102	CS3320	Spring	97	Knuth
112	MA2410	Fall	97	Chang
119	CS1310	Fall	97	Anderson

GradeReport

StudNo	SessIdent	Grade
17	112	14
17	119	12
8	85	16
8	92	16
8	102	14

4

Important Functions on a Database

- **Structure definition**: declare 5 files or relations + data types, e.g. Student(StudName, StudentNo, Class, Dept)
- **Population**: input data about specific students, courses, prerequisites, . . .
- **Querying**
  - ◇ Which are the prerequisites of the Database course ?
  - ◇ List students who got grade 14 or 16 for the Database course in 1993
- **Reporting**: prepare diplomas, with standard text, interspersed with name of student, courses taken, name of degree, grades, etc.
- **Modification, update of population**
  - ◇ Create a new session for the Database course
  - ◇ Enter a grade 16 for Smith in the Database Session
- **Modification of structure, of schema**
  - ◇ Create a new relation for instructors
  - ◇ Add Address attribute to relation Student

5

Transient and Persistent Data

- In practice, information systems often require persistent data
- Data: relevant facts about the domain of interest
  - ◇ **persistent**: continues to exist even when the system is not active
  - ◇ **transient**: created while an application is running and not needed when the application has terminated
- Persistent data must be stored in secondary memory (not just in computer memory) and organized to be made available to several applications

6

Data and Database Schema

- **Fundamental hypothesis of database modeling**: the information contained in a database is represented on two levels: (1) **data** (large, frequently modified) and (2) **structure of data** (small, stable in time)
- **Database schema**: description of DB structure, accessible by programs

$$\text{Database} = \left\{ \begin{array}{l} \text{Data Type} \\ \text{Metadata} \\ \text{Structure} \\ \text{Schema} \\ \text{Intension} \\ \text{Catalog} \\ \text{Directory} \\ \text{Data dictionary} \end{array} \right\} + \left\{ \begin{array}{l} \text{Instances} \\ \text{Occurrences} \\ \text{Data} \\ \text{Extension} \\ \text{Population} \end{array} \right\}$$

- **DBMS software is application-independent** $\Rightarrow$ it consults the database structure in the data dictionary to understand and execute application programs

7

- **Ontology** is another more recent term for designating the structure of an application domain (= schema information valid for several related applications)

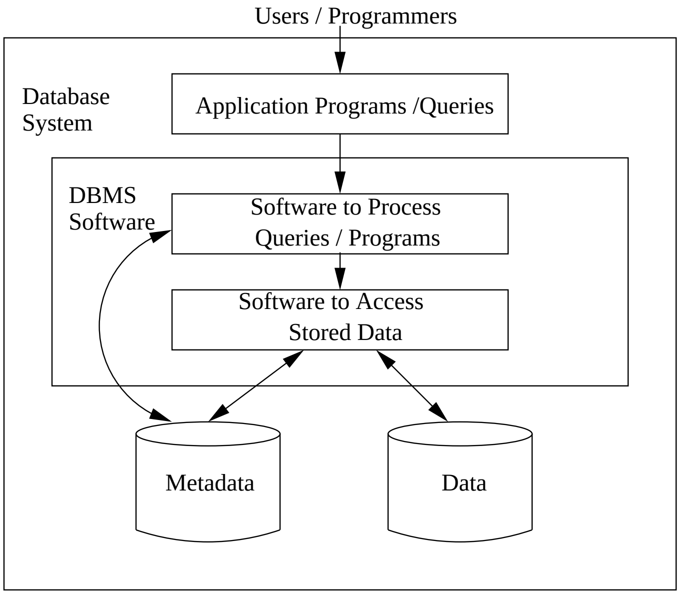
Various Levels of Schema

- A DBMS provides users with
  - ◇ a **conceptual representation** of information from the point of view of users
  - ◇ a **physical** or **internal representation** with the implementation details
- Both are necessary, but each has its purpose
- Users refer to the conceptual representation and the DBMS ensures the correspondence with the physical representation
- The success of relational technology has demonstrated that physical concepts can be hidden from users and that there are substantial advantages for doing so

8

- Technical vocabulary: **logical** versus **physical** concepts:
  - ◇ **logical** (old terminology) or **conceptual** (current terminology) information: deals with the user view on data (in terms of **concepts** familiar to actors in the application domain)
  - ◇ **physical** or internal concepts: concern the implementation of conceptual concepts into the hardware/software infrastructure (this is a technological area)

Gross Architecture of DB and DBMS Software



9

Data Structures in the Conceptual Schema

Student

StudName	StudNo	Class	Dept

Course

CourseName	CourseNo	Credits	Dept

Prerequisite

CourseNo	PrereqNo

Session

SessIdent	CourseNo	Semester	Year	Professor

GradeReport

StudNo	SessIdent	Grade

10

- In addition to data structures, the schema also comprises
  - ◇ the definition of domains for data elements (attributes)
  - ◇ the specification of constraints, to refine the data-structure part of the schema

Data Structures in the Physical Schema

- Student records are stored in a file as follows:

Data Item Name	Starting position	Length (bytes)
Name	1	30
StudentNumber	31	4
Class	35	4
Department	39	4

  - ◇ length of Student records = 42 bytes
  - ◇ the file is ordered by values of the Name field
  - ◇ the file is indexed on Name

11

Support of External Views

- A single database usually serves the needs of a community of users ⇒ different perspectives or **views** on the same data are often natural
- **View**: some subset of the database or some restructuring of the database suited for an application
- Views are redundant with the basic definition of the database (they may or may not be stored explicitly, this is an efficiency issue)
- DBMS takes care of the correspondence between views and database data
- View definitions are part of the database schema
- Other terms for view: subschema, external schema, derived relation

12

More on Views

- Relations in a database comprise **base relations** and **views** (users do not necessarily have to know which is which)
- Applications access the database through views, without necessarily knowing about the whole database
- Views may be
  - ◇ queried
  - ◇ combined in queries with base relations
  - ◇ used to define other views
  - ◇ in general, NOT updated freely

A View for Preparing Diplomas

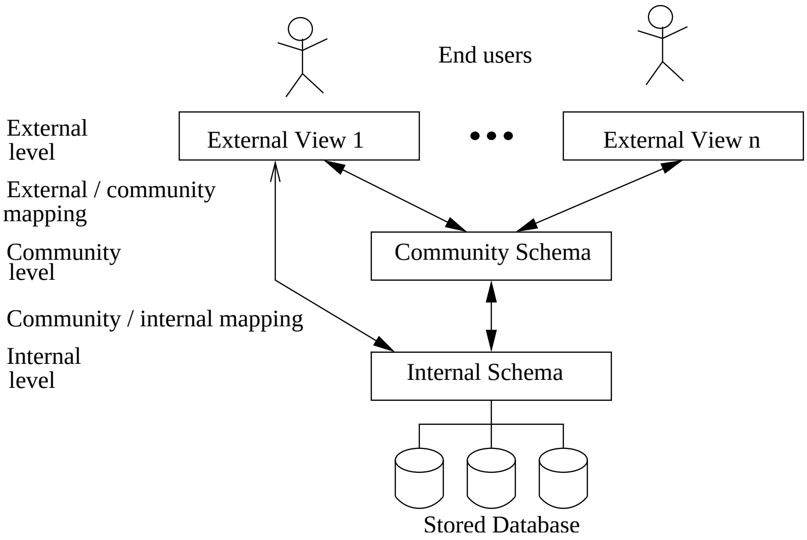
Diploma Data

StudName	StudentTranscript*			
	CourseName	Grade	Semester	Year
Smith	Introduction to CS	12	Fall	97
	Discrete Mathematics	14	Fall	97
Brown	Discrete Mathematics	16	Fall	96
	Introduction to CS	16	Fall	96
	Data Structures	14	Spring	97
	Database Management	16	Fall	97

13

- View Diploma Data is read-only (data cannot be entered into the database or modified or deleted thru Diploma Data)
- View definition, define view (new attr.) as SELECT ... (show SQL query)
- Display cross reference of views with base tables

Data Approach to DBMS Architecture: ANSI 3-Schema Architecture



14

- ANSI 3-Schema Architecture = general DBMS architecture to maintain several descriptions of the data in a database
- 3 levels of schemas
  - **external schemas** (user views) define the relevant data for application programs and hide the rest of the database
  - the **community schema** describes the common conceptual structure of the whole database; it contains and integrates the information contained in all the user views
  - the **internal schema** describes the database storage and access structures
  - data actually exists only at the internal level, it is accessed from the external level; DBMS provides **mappings** (compiled or interpreted) between levels, in both directions
- Note on vocabulary
  - in 1975, ANSI (American National Standards Institute) called **conceptual schema** what we call **community schema** here
  - nowadays, "conceptual" is more often used in another sense: a **conceptual schema** describes the user view of information, independently of the data model of the DBMS and its implementation
  - the conceptual schema is produced by the analysis phase in the process of database design

3-Schema Architecture and Data Independence

- **Data independence**: possibility to change the schema at one level without having to change it at the next higher level (nor having to change programs that access it at that higher level)
  - **logical data independence**: an external schema (and programs that access it) is insulated from changes that does not concern it in in the community schema (and in the physical schema)
  - **physical data independence**: the community and external schemas are insulated from changes in the physical schema

15

- With file processing, changes to file structure entail changes to application programs (e.g., COBOL mixes all three levels in its data division)
- Data independence did not come easy (one of the great debates of the relational "revolution" was about whether data independence could be realized with reasonable efficiency)
- **Program/data independence** is sometimes used instead of *data independence*, to emphasize that application programs remain unchanged when some changes are made to the data

Why Data Independence is Important
or Some Virtues of Abstraction

- It is a "divide-and-conquer" strategy to help **master complexity** and think precisely (user programs are more abstract, higher-level, simpler, and shorter)
- It leaves open more possibilities for the system to **optimize implementation strategies**
- It contributes to **stability in time** for applications (fewer changes in data structures to adapt to)

16

- Pre-relational (pre-historical) information systems went thru monolithic applications, combining in the same program user-interface management, implementation of business logic, data processing, and persistent-storage management
- The lack of data independence caused very high human cost in terms of re-programming, especially for evolution and maintenance, i.e., modifications to operational information systems for adapting to changes in requirements and changes in the system environment

Examples of Data (In)dependence

- Two fields within the same record: conceptual (relevant semantic link) or physical (clustering for fast joint access)
- A link between two records: conceptual (relevant semantic link) or physical (a pointer for fast navigational access)
- Ordering output data: how this is implemented (physical ordering or sorting) is left to the DBMS and invisible in application programs
- Adding a physical index to speed up an application should not require modifying the application program, the only visible effect will be efficiency
- Add a new field to a file: only programs that access the new information need be modified

17

Constraints

- **Integrity constraint**: any prescription (or assertion) on the schema (i.e., valid for all extensions of the database, now, in the past, and in the future)
- Constraints model extra information not definable in the data-structure part of the schema
- Constraints cannot be deduced from the database extension, they can only be verified, checked
- Examples
  - ◇ data types: grades are integers between 0 and 20
  - ◇ uniqueness of values: no two students have the same student number
  - ◇ referential integrity: all StudNo values in GradeReport must also appear as values of StudNo in Student

18

- *Consistency constraint* is a better term than *integrity constraint*
- Any piece of information can be given the status of constraint (i.e., belong to the schema); remember that
  - ◇ schema information is normally more stable in time than instance data
  - ◇ schema semantics is managed by DBMS software (still, not all constraints are managed by DBMS software, see later)

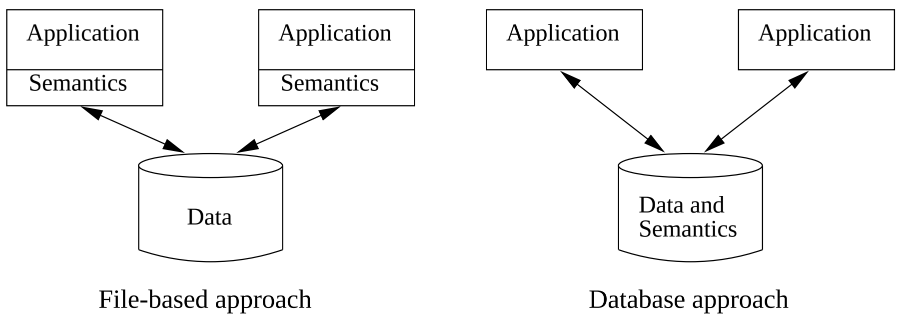
Constraints

- Some constraints **express a part of the general semantics of the data**, i.e., of the application domain modeled in the database
- Some constraints are technical (e.g., referential integrity is specific to the relational model)
- Not everything can be controled by constraints (e.g., misspellings in names can only be checked manually)
- Data models with richer, more expressive data structures have fewer explicit constraints than simpler ones (see later)
- Constraints have to be checked when updates are performed on the database

19

- **Basic rule of database modeling**:
  - ◇ the data in the database must conform to both the prescriptions of the database structure (the schema) and the constraints
  - ◇ it is often said that the constraints belong to the schema
  - ◇ basic modality: every piece of data that does not contradict the prescriptions of schema + constraints is acceptable in the extension of the database ("everything that is not explicitly forbidden is permitted")

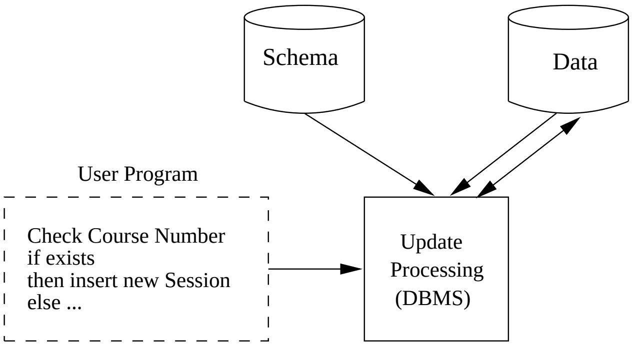
Constraints are Properties of Data, not of Applications



20

- A progressive evolution of DBMS technology from traditional file processing is to move constraints as much as possible from application programs into the schema

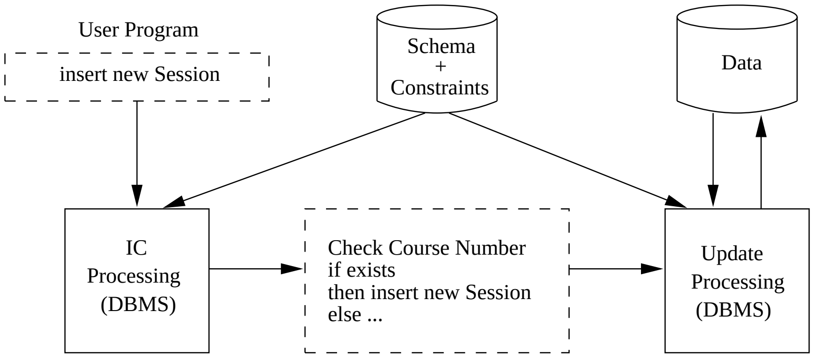
Constraint Enforcement in Application Programs



21

- **Example**: application program that adds a new Session for a Course in the presence of a referential constraint, that requires that the CourseNo must exist
- The program is given as input a 5-tuple of values: (SessIdent, CourseNo, Semester, Year, Professor)

Constraint Enforcement by DBMS



- DBMS enforcement of constraints ⇒ simplification of user programs at the cost of more complexity for DBMS software

22

Various Semantics in Modeling

(1) **Schema semantics**: used and maintained by system software (data structures, consistency constraints)

(2) **Instance semantics**: informal, intuitive real-world semantics; not used nor maintained by system software

(3) **Denotation semantics**: relates elements in (1) with corresponding elements in the real world; makes explicit the relationship between (1) and (2)

(4) **Environment semantics**: relates elements in (1) and their corresponding real-world elements with system environment in the real world

- (3) and (4) are usually informal documentation in the data dictionary

23

- In the database approach, the schema expresses the **general semantics** of the part of the world that is modeled (in terms of data structures, concepts, categories, constraints) while the data expresses the semantics of individual objects
- The **instance semantics** is not modeled nor exploited by the tools (i.e., the DBMS): it is left to the interpretation of users, possibly helped by programs (e.g, to consult an informal definition)
- Another part of the semantics deals with the **correspondence between schema** and **real world**. It is rarely exploited by tools and is at best represented informally in design documents and in the **data dictionary** of the database (documentation)
- **General semantics** (in the schema) comprises
  - ◇ the **data structures** for the application domain in the data model of the DBMS
  - ◇ classical **constraints** expressing regularities in the application domain (e.g., uniqueness, keys, referential integrity, normal forms)
  - ◇ ad-hoc constraints (e.g., there are no two different sessions of the same course with the same professor, or, all CS students take at least 3 CS courses)
- Extracting the relevant general semantics of an application domain is the process of **database design**; it results in a conceptual schema which is the basis of an agreement between domain analysts and users, as well as the starting point for implementing the database

Data Model

- Data model = languages for defining structure and behavior
  - ◇ **structure**: data types, relationships, constraints
  - ◇ **behavior**: basic operations for retrievals and updates
- DBMSs support several levels of data models:
  - ◇ **conceptual**: closer to user view on data (nowadays, mostly entity-relationship)
  - ◇ **physical**: for storage structures and access methods
  - ◇ **logical** or **implementation**: (historical) compromise between conceptual and physical organization
- **Object-oriented models** allow to integrate user-defined structure and behavior (but object orientation in the database world is complicated)

24

- The relational model was the first model to be defined: the concept of data model is an important contribution of the relational era

- Although it clearly distinguishes between information content from the user point of view and its implementation, the relational model cannot be considered as a "conceptual" model
- More "semantic" data models are appropriate as conceptual models for database design (currently, typically, the entity-relationship model)
- The relational model has become a compromise: logical relational schemas (e.g., in SQL) give a simple view of data as relations (or tables, or abstract files)
- The implementation models of commercial DBMSs are mostly relational, although network and hierarchical systems are still used
- Network and hierarchical "models" were defined after the fact, as more or less satisfactory a posteriori abstractions

Data Models and their Implementation

- **Data model**:
  - ◇ abstract, self-contained mechanisms for defining data structures and operations
  - ◇ hides low-level storage details ⇒ abstract machine for user interaction
- **Implementation** of the data model: physical realization on a computer architecture
- The distinction relates to physical data independence (distinction between logical and physical concepts)

25

Database Languages

- **Data Definition Language (DDL)**, for
  - ◇ writing all schemas and mappings between schemas
  - ◇ specifying constraints
- **Data Manipulation Language (DML)**, for
  - ◇ database manipulation (retrieval, insertion, deletion, modification) with query languages and programming languages
  - ◇ user-friendly interfaces (graphical, menu-based, forms-based, natural language, parametric)

26

Levels of DML Languages

- **High level** (e.g., SQL)
  - ◇ specify **what** data is to be retrieved rather than **how** to retrieve it
  - ◇ used on their own or embedded (**data sublanguage**) in a programming language (**host language**) like C, Pascal, COBOL
  - ◇ also called **declarative**, assertional, nonprocedural, set-at-a-time, set-oriented
- **Low level**
  - ◇ retrieve individual records and process each one separately
  - ◇ also called **procedural**, or record-at-a-time, navigational

27

Database Functions and Application Functions

- **Database functions** or **DBMS functions**: supplied by the DBMS and invoked in application programs
- **Application-program functions**: to be programmed in application programs
- Evolution:
  - ◇ the power of DBMS software is continuously increasing
  - ◇ more and more functions that used to have to be programmed are progressively turned into DBMS functions, because
    - ∗ some data-management issues are better understood and can be turned into DBMS modules
    - ∗ the processing ressources continuously increase

28

Other DBMS Functions

- Concurrency control
- Backup and recovery
- Redundancy management
- Access control
- Performance optimization
- Metadata management
- Active features (rules, triggers)

29

DBMS Function: Concurrency Control

- **Transaction processing** (OLTP) applications (e.g., banking and airline systems), with multiple simultaneous users
- **Concurrency control**: ensures correctness of competing accesses to same data
- Correctness: 4 desirable properties (A.C.I.D.)
  - ◇ **Atomicity**: "all or nothing", transactions execute entirely or not at all
  - ◇ **Consistency**: transactions move the DB from a consistent state to a consistent state
  - ◇ **Independence or isolation**: no partial effects of incomplete transactions are visible
  - ◇ **Durability**: successfully-completed transactions are permanent, cannot be undone

30

- Transaction = one execution of a user program (executing the same programs several times corresponds to several transactions)
- Transaction = basic unit of change as seen by the DBMS
  - ◇ partial transactions are not allowed (atomicity)
  - ◇ the effect of a collection of transactions is equivalent to some serial execution of the transactions (serializability)

Three Transactions

T_1 Read NP
NP ← NP-1
Write NP

T_2 Read NP
NP ← NP-1
Write NP
Read NQ
NQ ← NQ+1
Write NQ

T_3 Read NP
...
Read NQ
...
Read NP

31

- T_1 could be the reservation of an instance of a resource, e.g., a seat for a particular flight; NP is then the number of seats available on that flight; T_2 is the canceling of a reservation (NQ) combined with a reservation (NP); T_3 just queries the database
- Read NP is a transfer of information from the database to the user space
- NP ← NP - 1 is performed in the user space (i.e., with no effect on the database)
- Write NP modifies the database

An Incorrect Schedule

Step	T_1	T_2	T_3
1		Read NP	
2	Read NP		
3		NP $\leftarrow$ NP-1	Read NP
4		Write NP	
5		Read NQ	
6	NP $\leftarrow$ NP-1		
7	Write NP		Read NQ
8		NQ $\leftarrow$ NQ+1	
9		Write NQ	Read NP

32

- Both T_1 and T_2 work with the same value from the database
- The update by T_2 is not preserved in the database

A Correct Schedule

Step	T_1	T_2	T_3
1		Read NP	
2		NP ← NP-1	Read NP
3		Write NP	
4	Read NP		
5	NP ← NP-1		
6	Write NP		Read NQ
7		Read NQ	
8		NQ ← NQ+1	
9		Write NQ	Read NP

33

- Correctness is obtained by sequencing potentially conflicting updates

Which Level of Concurrency Control?

- Tradeoff between
  - ◇ **efficiency** (# transactions/sec), and
  - ◇ **cost** of maintaining consistency
- Complex, powerful piece of engineering (particularly for geographically distributed and physically redundant databases) coupled with fine management optimization
- Example: how much do we want to protect against
  - ◇ two simultaneous withdrawals from the same bank account
  - ◇ multiple reservations of the same airplane seat

34

DBMS Functions: Backup and Recovery

- DBMSs provide facilities for recovering from hardware and software failures
- If the computer system fails during a complex update program
  - ◇ the DB must be restored to its state before the program started, or
  - ◇ the program must be resumed where it was interrupted so that its full effect is recorded in the database
- More complex and important in a multi-user environment

35

DBMS Function: Redundancy or Replication Management

- **Redundancy**: storing several copies of the same data
- Frequent in traditional file processing: a goal of the database approach was to control redundancy as much as possible
- Problems with redundancy
  - ◇ waste of storage space
  - ◇ duplication of effort to perform a single conceptual update
  - ◇ danger of introducing inconsistency if multiple updates are not coordinated
- Replication of the same data may be useful for optimizing physical accesses (typically in distributed databases)

36

- Controling redundancy was a major progress of database technology over file systems
- If the right decisions can be made during database design, the resulting central database schema presents a more uniform data representation (e.g., avoids different types for the same data as typically happens with file systems)
- If present for efficiency, redundancy should remain invisible to ordinary users and be under the control of the DBMS (a complex technical problem in general)

DBMS Function: Access Control

- Who accesses what data, to do what, when, from where, etc.
- Access control is mandatory in a multiuser database, e.g., for confidentiality
- Various access modes to data (e.g., read only, read and update)
- DBMS subsystem enforces **security and authorization**
- Restrictions concern **programs** (e.g., who can create new bank accounts) and **data** (e.g., which bank accounts can I see)

37

- The data dictionary holds information about users and their access privileges (e.g., name and password)
- Several levels of access privileges
  - ◇ to create a database
  - ◇ to authorize (grant) additional users to
    - ∗ access the database
    - ∗ access some relations
    - ∗ create new relations
    - ∗ update the database
  - ◇ to revoke privileges

DBMS Function: Performance Optimization

- Good manual optimization of DB programming is scarce and expensive
- Performance optimization is largely a DBMS function
- This is made possible by
  - ◇ physical data independence
  - ◇ high-level data models with user programs that can be optimized by DBMS software (unlike navigational record-at-a-time programs for which optimization can only be manual, i.e., left to users)
- DBMS maintains information (metadata) on database populations, in addition to storage structure (conceptual schema) and access paths (physical schema)
- Actual optimum varies with evolution of DB population: physical reorganizations are sometimes necessary (e.g., add index, drop index, sort file)

38

- Efficient relational query optimization was a key to the acceptance of the relational model in the 80's
- What matters today is human performance, not machine performance

DBMS Function: Metadata Management

- Data about data is also data: **metadata**
- System catalog (or data dictionary): special DB maintained by DBMS
- Information in the catalog: data objects, DB statistics, physical structures and access paths, user access privileges, etc.
- Accessible to DBMS functions and to users

39

Example Catalog of Relational Schema

RelationAttributesCatalog

RelName	AttrName	AttrType	FKMember	FKRelation
Employee	SSN	String(9)	no	
Employee	FName	String(15)	no	
Employee	MInit	char	no	
...	...	...	...	...
Department	DName	String(10)	no	
Department	DNumber	Integer	no	
Department	MgrSSN	String(9)	yes	Employee
...	...	...	...	...

RelationKeys

RelName	KeyNumber	MemberAttr
Employee	1	SSN
Department	1	DNumber
Department	2	DName
...	...	...

40

Example Catalog of Relational Schema

RelationIndexes

RelName	IndexName	MemberAttr	IndexType	AttrNumber	AscDesc
WorksOn	ESSNIndex	ESSN	clustering	1	Asc
WorksOn	EPIndex	ESSN	secondary	1	Asc
WorksOn	EPNIndex	PNO	secondary	2	Asc
...	...	...	...	...	...

ViewQueries

ViewName	Query

OldEmps	`Select SSN, Fname,LName`
	`From Employee`
	`Where BDate < 01/01/1950`

...	...

ViewAttributes

ViewName	AttrName	AttrNumber
OldEmps	SSN	1
OldEmps	FName	2
OldEmps	LName	3
...	...	...

41

Active-Database Technology

- **Passive DBMS**: all actions on data result from explicit invocation in application programs (they only do what application programs tell them to do)
- **Active DBMS**: execution of actions can be automatically triggered in response to monitored **events**, including
  - ◇ database updates: upon deletion of the data about a customer
  - ◇ points in time: on January 1, every hour
  - ◇ events external to the database: whenever paper jams in the printer

42

- Evolution of database technology has been going thru representing and supporting more functionality of database applications within the DBMS, e.g.,
  - ◇ checks of some types of **integrity constraints** (produced from a declarative definition located with the database schema)
  - ◇ **stored procedures**: precompiled procedures located within the database, invoked from application and system programs
  - ◇ **common semantics** abstracted from application domains (e.g., for spatial, multimedia, temporal, deductive, active databases)
- **Active-database technology**
  - ◇ a relatively recent extension of traditional DBMS technology
  - ◇ most commercial RDBMSs include some capability for rules or **triggers**
  - ◇ research prototypes provide more comprehensive support for active rules than RDBMSs
- Application semantics in programs for active DBMSs is expressed in:
  - ◇ traditional application programs (as for passive DBMSs)
  - ◇ rules (in the database, available to all applications)

Event - Condition - Action Rules

- When an event occurs, if a condition holds, then an action is performed

> Event
>   a customer has not paid 3 invoices at the due date
> Condition
>   if the credit limit of the customer is less than 20 000 Euros
> Action
>   cancel all current orders of the customer

- ECA rules are part of the database (⇒ “rule base”), available to all applications

43

Rules May Express Various Aspects of Application Semantics

- **Static constraints** (e.g., referential integrity, cardinality, value restrictions)
  - only regular students can register at the library
  - students can register in no more than 20 courses
  - the salary of employees cannot exceed the salary of their manager
- **Control, business rules, workflow management**
  - when data for new students is recorded, data is automatically entered to register the students in the mandatory courses
  - all expenses exceeding 50K must be approved by a manager
  - when an order has been accepted, an invoice is sent
- **Historical data**
  - the data about completed orders is transferred monthly to the data warehouse

Semantics Modeled by Rules (cont'd)

- Implementation of **generic relationships** (e.g., generalization)
  - ◇ a person is a student or a lecturer, but not both
- **Derived data**: materialized attributes, materialized views, replicated data
  - ◇ the number of students registered in a course must be part of the course data
  - ◇ orders received are summarized daily in the planning database
- **Access control**
  - ◇ employees can view data about their own department only
- **Monitoring**: performance, resource-usage monotoring
  - ◇ the number of disk accesses of each database query is recorded and statistics are produced weekly
  - ◇ each access to our web pages is reflected in the usage database

45

- **Exercise**: rephrase the above examples as event-condition-action rules
- Note that many examples have a more declarative form than ECA rules

Benefits of Active Technology

- **Simplification of application programs**: part of the functionality can be programmed with rules that belong to the database
- **Increased automation**: actions are triggered without direct user intervention
- **Higher reliability** of data thru more elaborate checks and repair actions ⇒ better computer-aided decisions for operational management
- **Increased flexibility** thru centralisation and code reuse ⇒ reduced development and maintenance costs

46

People around DBs/DBMSs

- **Casual users**
- **Parametric users**
- **Application programmers**
- **Database designers**
- **Database administrator** (DBA)
- **DBMS vendors**
- **System programmers**
- **Operators**

47

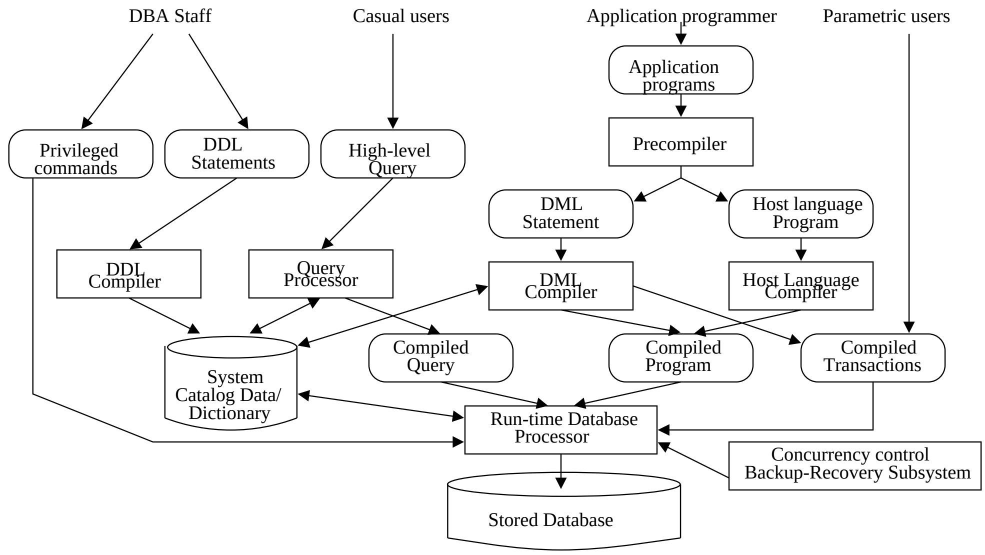
People and Functions around DBs/DBMSs

- End users:
  - **casual users**: occasional unanticipated access to DB (e.g., tourists, managers)
  - **parametric users**: query and update the database through fixed programs (invoked by non-programmer users) (e.g., banking)
- **Application programmers**: implement database application programs that facilitate data access for end users
- **Database designers**:
  - prepare external schemas for applications
  - identify and integrate user needs into a conceptual (or community, or enterprise) schema
- **Database administrator** (DBA):
  - define the internal schema, defining subschemas (with database designers), and specify mappings between schemas
  - coordinate, supervise, and monitor database usage
  - supervise DBMS functions (e.g., access control, performance optimization, backup and recovery policies, conflict management)
- **DBMS vendors** and their technical staff (build and maintain the DBMS software)
- **System programmers**: interact with DBMS software and internal database level
- **Operators**: responsible for running and maintaining the HW/SW for the DBMS, backup, recovery from failures, etc.

Points of View on DBMS Architecture

- **Data**: several data views and their relationships (ANSI 3-schema architecture)
- **Components**: DBMS software viewed as a number of components providing functionality; emphasis on DBMS system design and implementation
- **Functions**: different classes of users and functions performed for them by DBMS software; no emphasis on how functions are realized
- **Operation**: how DBMS functions are realized with current software, hardware, and network infrastructure (client-server architecture)

Main DBMS Components



49

- DDL compiler: builds inter-schema mappings
- Application program = mixture of regular program and DML statements (SQL queries)
- High-level queries ≈ DML statements
- Compiled program ≈ compiled transaction
- Not shown: answers to queries and programs, access control, ...

Advantages of the Database Approach

- Summary
  - ◇ separate **DBMS functions** from **application functions**
  - ◇ move application domain semantics out of programs into DB schema
- **Reduced application-development time**
  - ◇ simpler programs because many functions can be invoked from the DBMS
- **Uniformization** of organizational procedures
  - ◇ at the expense of more effort on initial database design
- **Reduction of redundancies** in personnel and procedures
  - ◇ e.g., to manage data redundancy or complex conversion processes
- **Centralized management** of information, performance, conflict resolution
- **Rationalization** of information processing
  - ◇ users can concentrate on using information
- **BUT** DBMS software is complex and expensive, not all applications need all DBMS functions

50