+

CSIT6000P Spatial and Multimedia Databases

2022 Spring

RDBMS Review

Prof Xiaofang Zhou

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

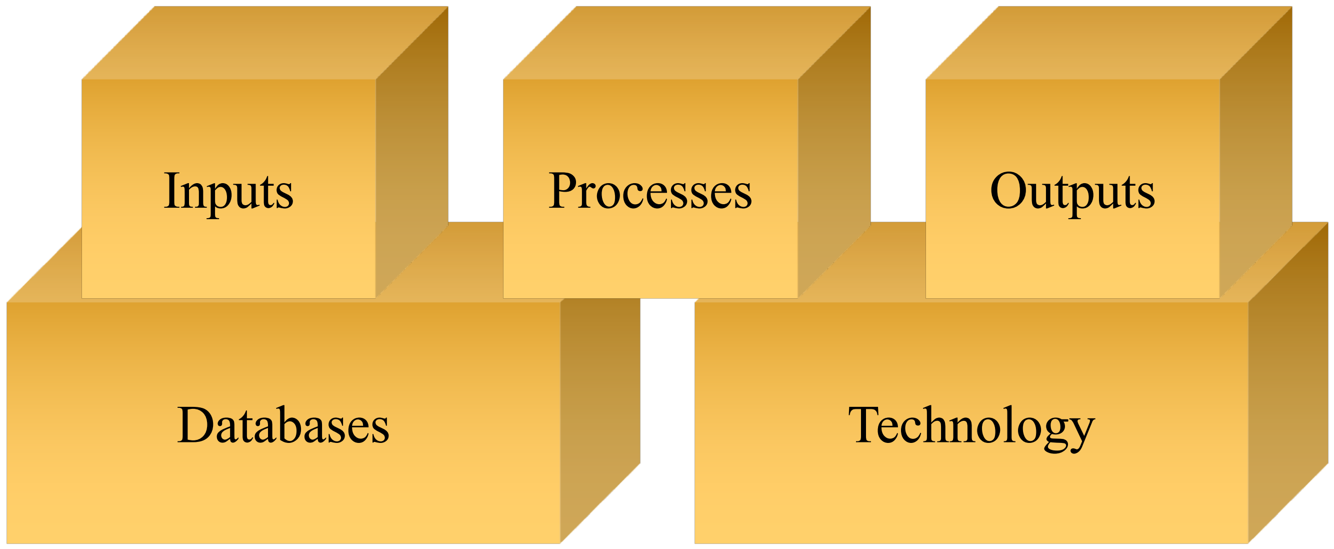
+ Data

- Data: facts and statistics collected for reference or analysis, typically digitised
- Data is the new fuel for the global economy
- *Store it, find it and use it*

The Economist
Theresa May v Brussels
Ten years on: banking after the crisis
South Korea's unfinished revolution
Biology, but without the cells
MAY 6TH–12TH 2017
The world's most valuable resource
amazon
Google
Data and the new rules of competition

+ Information Systems

- A system that manages information
- Key building blocks:



+ Data Management

- Data Management is an essential skill for future workforce. It can be used to capture, store, retrieve, analyze, present and interpret (large amount of) data!
- **Banking** is an example of an *application area* where data plays a central role

...think of another example and discuss with the person next to you how data plays a central role in your example application

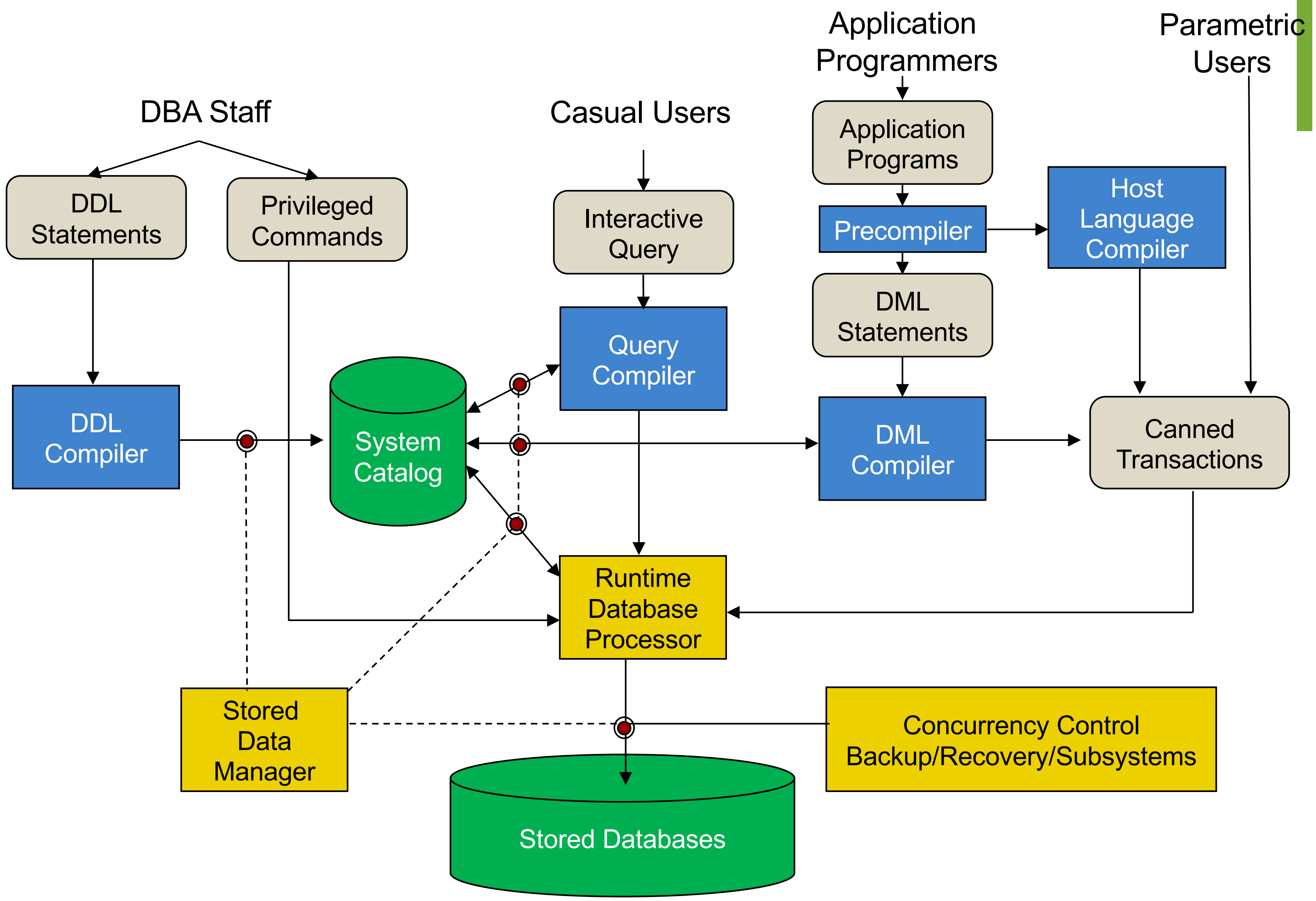
+ What is a DBMS?

- DBMS - DataBase Management System
- Multiple-billion market for DBMS products and services!
- A software system for defining, constructing and manipulating databases for various applications
- DBMS may be general purpose (business applications) or special purpose (biological databases, geographic information, ...)

+ The DBMS Facilitates ...

- Defining a database
  - Specifying the types, structures, and constraints for the data
- Constructing a database
  - Storing the data on a storage medium
- Manipulating a database
  - Querying and updating the database
- Maintaining a database
  - Ensuring database efficiency, correctness and safety

+ The DBMS Software



+ Typical Functions of the DBMS

- Controlling redundancy
- Restricting unauthorized access
- Providing multi-user interfaces
- Representing complex relationships
- Enforcing integrity constraints
- Providing backup and recovery

+ Controlling Redundancy

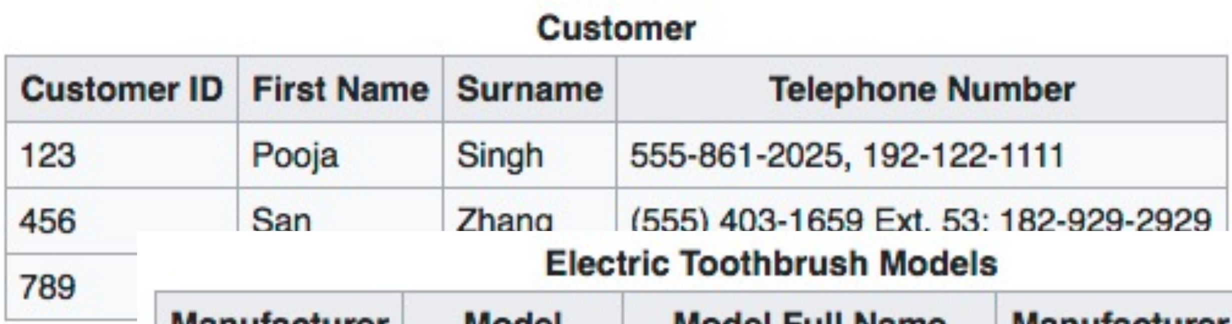
- Redundancy occurs when one fact is stored in more than one place
- Redundancy can cause
  - Duplication of effort
  - Waste of storage space
  - Inconsistent data

Stud-No	Name	Degree	Subject	Grade
90	Smith	BA	CS182	7
87	Brown	BA	CS182	7
98	James	BSc	CS181	6
90	Smith	BEng	CS181	6

Stud-No	Name	Degree	Finance-Type
90	Smith	BEng	Self
87	Brown	BA	Scholarship
98	Harrison	BSc	Self

..however, redundancy sometimes enhances performance - DBMS provides an environment where redundancy can be controlled

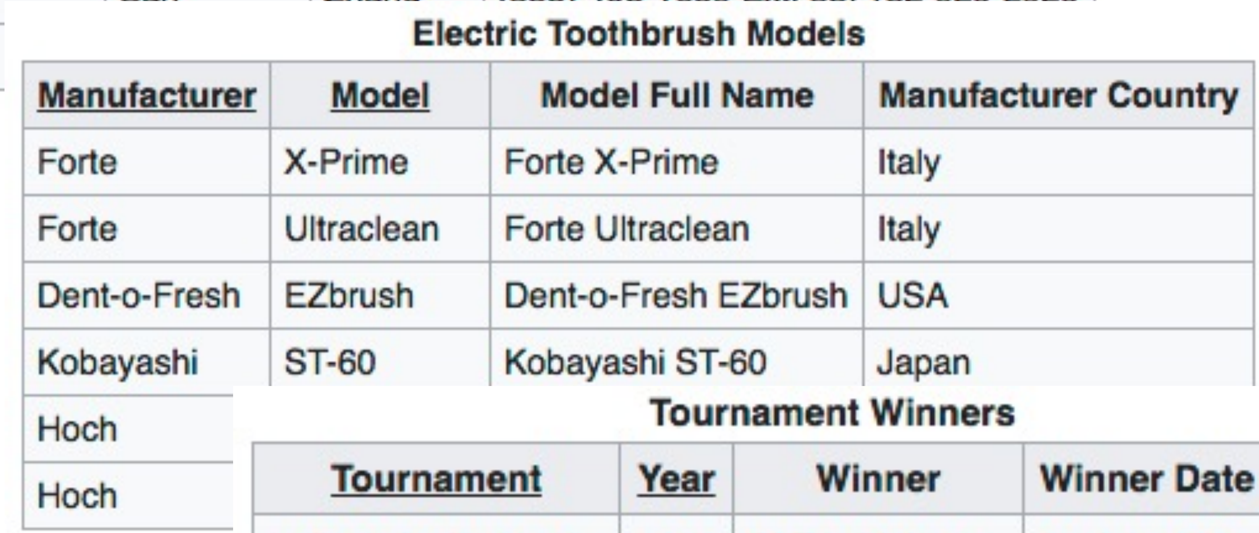
+ Example: Normal Forms



Customer

Customer ID	First Name	Surname	Telephone Number
123	Pooja	Singh	555-861-2025, 192-122-1111
456	San	Zhang	(555) 403-1659 Ext. 53; 182-929-2929
789			

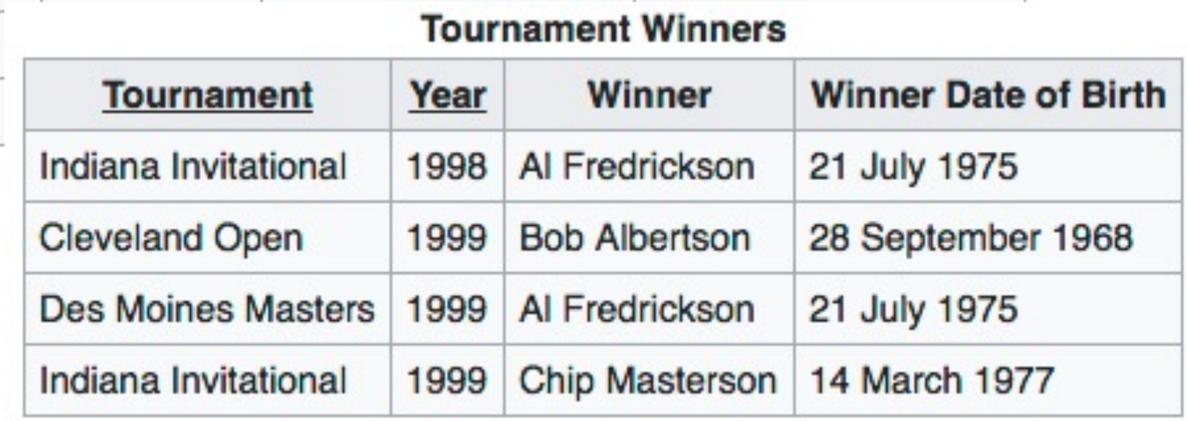
1NF?



Electric Toothbrush Models

Manufacturer	Model	Model Full Name	Manufacturer Country
Forte	X-Prime	Forte X-Prime	Italy
Forte	Ultraclean	Forte Ultraclean	Italy
Dent-o-Fresh	EZbrush	Dent-o-Fresh EZbrush	USA
Kobayashi	ST-60	Kobayashi ST-60	Japan
Hoch			
Hoch			

2NF?



Tournament Winners

Tournament	Year	Winner	Winner Date of Birth
Indiana Invitational	1998	Al Fredrickson	21 July 1975
Cleveland Open	1999	Bob Albertson	28 September 1968
Des Moines Masters	1999	Al Fredrickson	21 July 1975
Indiana Invitational	1999	Chip Masterson	14 March 1977

3NF?

+ Restricting Unauthorized Access

- Different user groups may have different access privileges (Create/Alter, Update, and Retrieve), which are controlled through DBMS security sub-system, through the use of Accounts & Passwords
- Casual users may not have access to confidential data, e.g medical records, salary packages, police reports
- Parametric users may be given update access, but are generally not allowed to change the structure of data
- Database administrators (DBAs) generally have highest privileges, create user accounts and enforce restrictions

+ Providing Multi-user Interfaces

- Query languages for casual end users
- Programming language interfaces for application programmers
- Forms and commands for parametric users

- Other types of interfaces
  - Graphical User Interfaces (GUI)
  - Interface for Web enabling
  - Natural language interfaces…

```
Compusoft <R> MS DOS
<C> Copyright Compusof
C:\>_
```

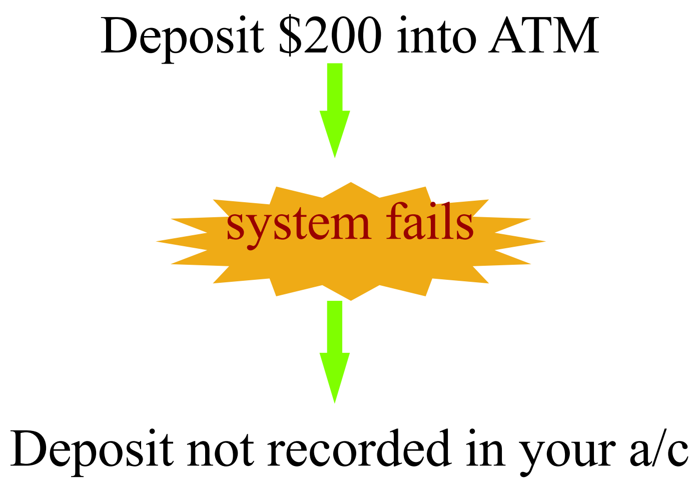
+ Enforcing Integrity Constraints

- The DBMS has the capability to define and enforce integrity constraints which are restrictions placed on the data, based on the semantics or meaning of the data
  - Every subject must have a unique code
  - A student cannot have 2 different grades for the same subject
  - A student cannot enroll in more than four 12-credit subjects in a semester
  - Student-No must be a 9-digit integer

  - Generic constraints vs application-specific constraints
  - DBMS cannot check spelling or typing errors, for example, if 5 was entered as the grade of a student getting 7 - DBMS will not identify the error!

+ Providing Backup and Recovery

- DBMS provides facility to recover from hardware and software failures through its backup and recovery sub-system
  - An update program is executing
  - Computer System fails in the middle of the update
  - DBMS restores the database to a state prior to the update and restarts the update program



+ Question:

- Which of the following is not a function of the DBMS

  A. Enforcing integrity constraints
  B. Design a database
  C. Backup and recovery of database
  D. Providing secure access to the database

+ Database System Components

- The Stored Database
  - A collection of related facts
- The DBMS
  - The software that defines, constructs and manipulates a database
- The Applications
  - The programs (in specific languages) that manipulate the database
- The Users
  - People who use the database system, through the DBMS interface or through application programs

+ Users of the Database System

- Database Administrators
- Database Designers and Application Programmers
- End Users
  - Casual End Users
  - Parametric End Users
  - Sophisticated End Users

+ Why Using a Database System?

- The database approach provides a central store of data and meta-data, and thus
  - It is not internal to an application program, as in traditional file processing environments
  - Provides shared access for multiple users
  - Relieves the application programmers from various tedious bookkeeping tasks
  - Provides the facility to change the data without affecting the applications

→ Provides Data Independence

+ Three-Schema Architecture

- Characteristics of the database approach:
  - Separation between programs and data
  - Support of multiple views of data
  - Use of a catalog to store schema
- To implement these characteristics, a “three-schema” architecture was proposed that provided
  - Logical Data Independence
  - Physical Data Independence

+ Categories of Data Models

Data model is an abstract view of data that excludes some details

More abstract (with less implementation details)

High-Level (Conceptual Schema)

Implementational (Logical Schema)

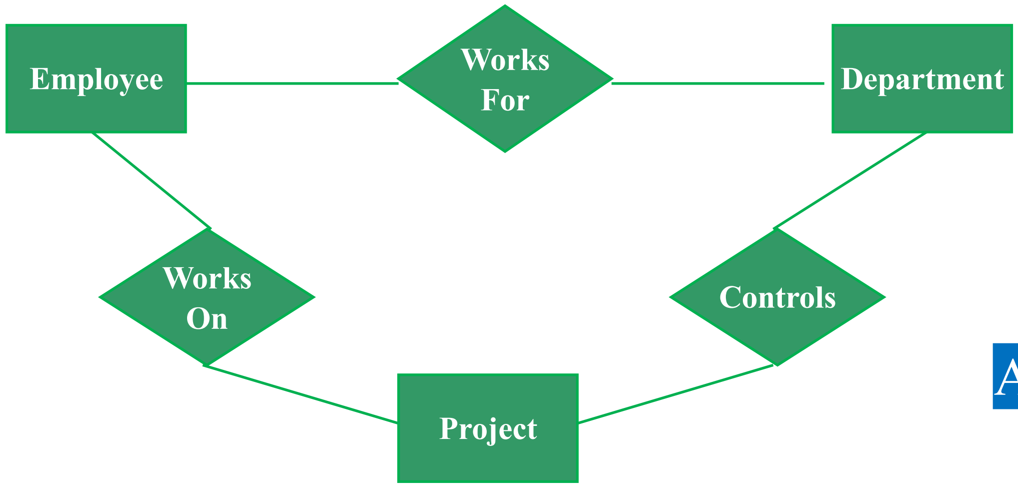
Low-Level (Physical Schema)

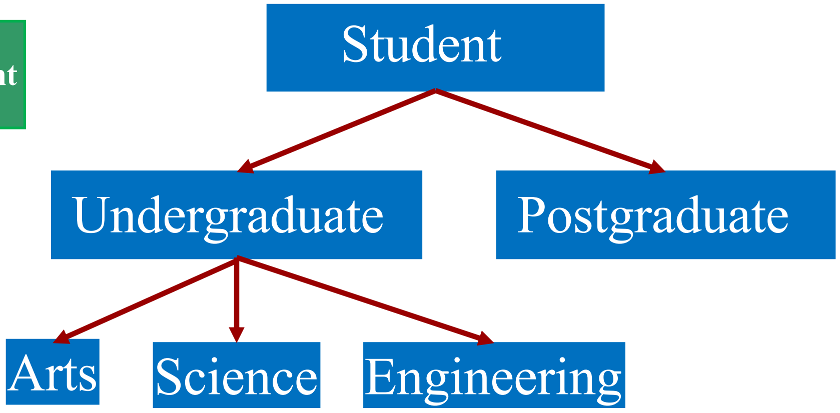
Less abstract (with more implementation details)

…external schema

+ High-Level (Conceptual)

- Provides concepts close to the way users perceive data
- Often expressed with a graphical notation, e.g. ER or UML
- No detail about computer implementation, storage or system





+ Implementational (Logical)

- Provides concepts understood by users, but reflect data organization
- Hides some details of data storage
- An implementation model as given by the DBMS is used e.g. Relational Model

EMP_NAME	EMP_ADDRESS	DEPARTMENT
Nicole Smith	1 Pine Road	Info. Systems
Joe Bates	32 Chandler Rd	Manufacturing

+ Low-Level (Physical)

- Describes how data is represented and organized on the computer's storage devices (e.g., disks)
- The database specialist uses the DBMS's DDL (Data Definition Language) to communicate the specification to the DBMS
- These models are usually only viewed by database specialists, e.g., DBAs (Database administrators)

+ Data Independence

- Logical independence
  - To allow independent changes on logical schema without having to change conceptual scheme or the application programs that access the database via the external schemas
- Physical independence
  - To change internal schema without having to change the logical, conceptual or external schemas

+ Example: Logical Data Independence

EMP_NAME	EMP_ADDRESS	DEPARTMENT
Nicole Smith	1 Pine Road	Info. Systems
Joe Bates	32 Chandler Rd	Manufacturing

+ A User's View

EMP_NAME	EMP_ADDRESS	DEPARTMENT
Nicole Smith	1 Pine Road	Info. Systems
Joe Bates	32 Chandler Rd	Manufacturing

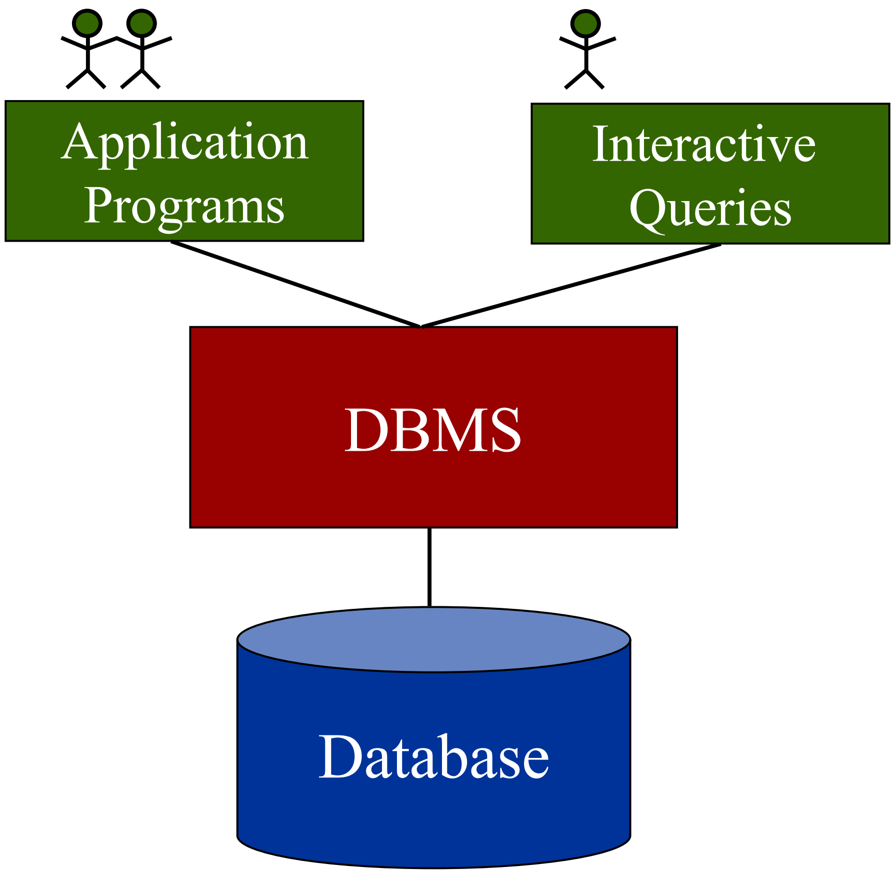
+ Conceptual Schema Enlarged

<u>EMP_NAME</u>	EMP_ADDRESS	DEPARTMENT	PHONE
Nicole Smith	1 Pine Road	Info. Systems	345671
Joe Bates	32 Chandler Rd	Manufacturing	54635

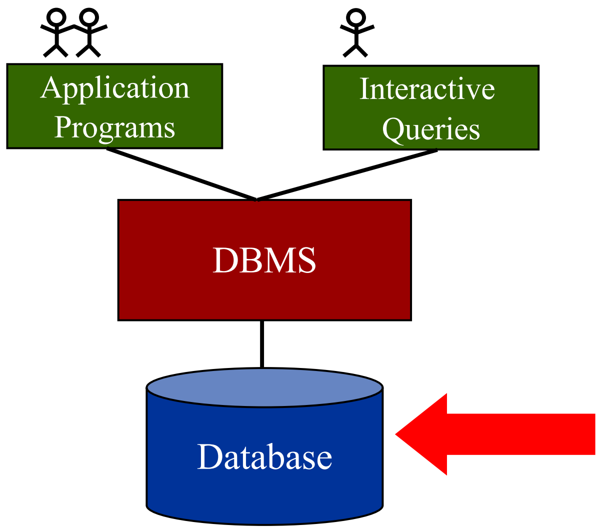
+ User’s View Unchanged

EMP_NAME	EMP_ADDRESS
Nicole Smith	1 Pine Road
Joe Bates	32 Chandler Rd

+ A Database System

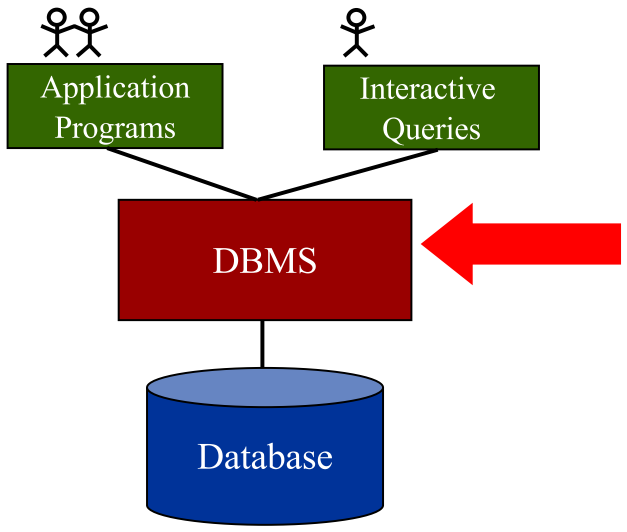


+ The Database



- Database Design
  - Entity Relationship diagrams, Functional dependencies, Normalisation
- Data Models
  - Relational, Object Oriented, Object Relational, Network, Hierarchical
- Physical Storage
  - Organisation, Hashing, Indexing

+ The DBMS

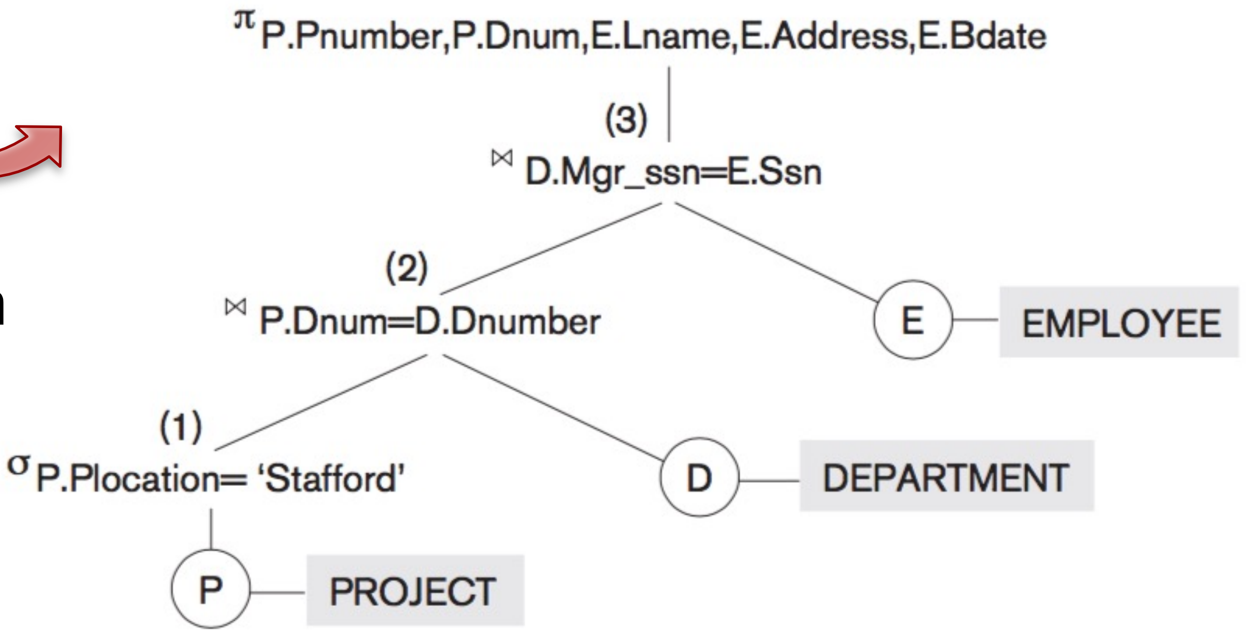


- System Catalogue
- Query Processing
  - Interactive queries, Optimization
- Transaction Processing
  - Failures, Schedules, Recoverability, Serializability
- Concurrency Control & Recovery
  - Locking, Time stamping, Recovery techniques
- Security Management
  - Access Control and Privileges

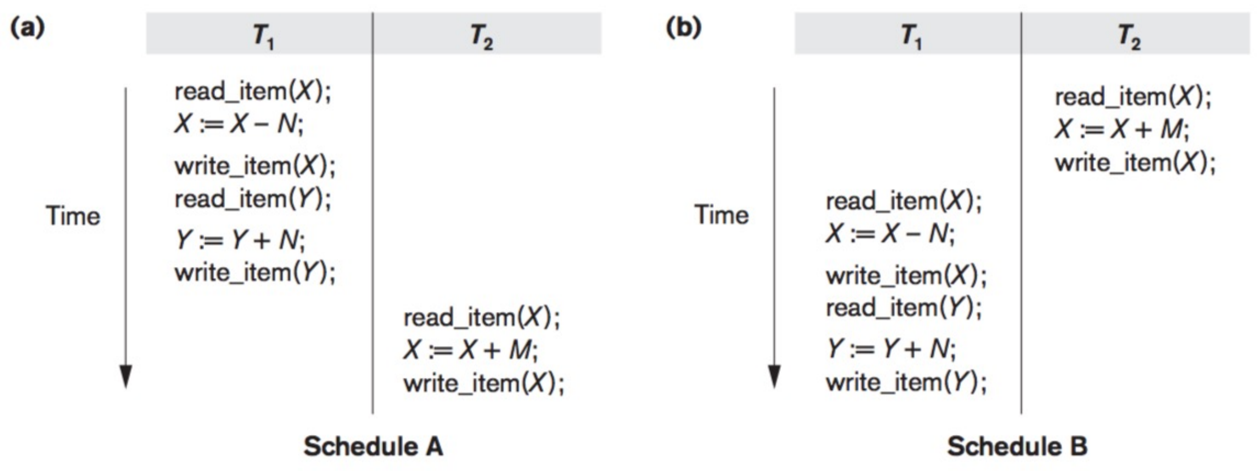
+ Example: Query Processing

```
SELECT   P.Pnumber, P.Dnum, E.Lname, E.Address, E.Bdate
FROM     PROJECT AS P, DEPARTMENT AS D, EMPLOYEE AS E
WHERE    P.Dnum=D.Dnumber AND D.Mgr_ssn=E.Ssn AND
         P.Plocation= 'Stafford';
```

- Parsing
- Optimisation
- Execution


$\pi_{\text{P.Pnumber,P.Dnum,E.Lname,E.Address,E.Bdate}}$
(3)
$\bowtie_{\text{D.Mgr\_ssn=E.Ssn}}$
(2)
$\bowtie_{\text{P.Dnum=D.Dnumber}}$
E
EMPLOYEE
(1)
$\sigma_{\text{P.Plocation= 'Stafford'}}$
D
DEPARTMENT
P
PROJECT

+ Example: Transaction Management



(a)

Time ↓	T_1	T_2
	read_item(X);	
	$X := X - N$;	
	write_item(X);	
	read_item(Y);	
	$Y := Y + N$;	
	write_item(Y);	
		read_item(X);
		$X := X + M$;
		write_item(X);

Schedule A

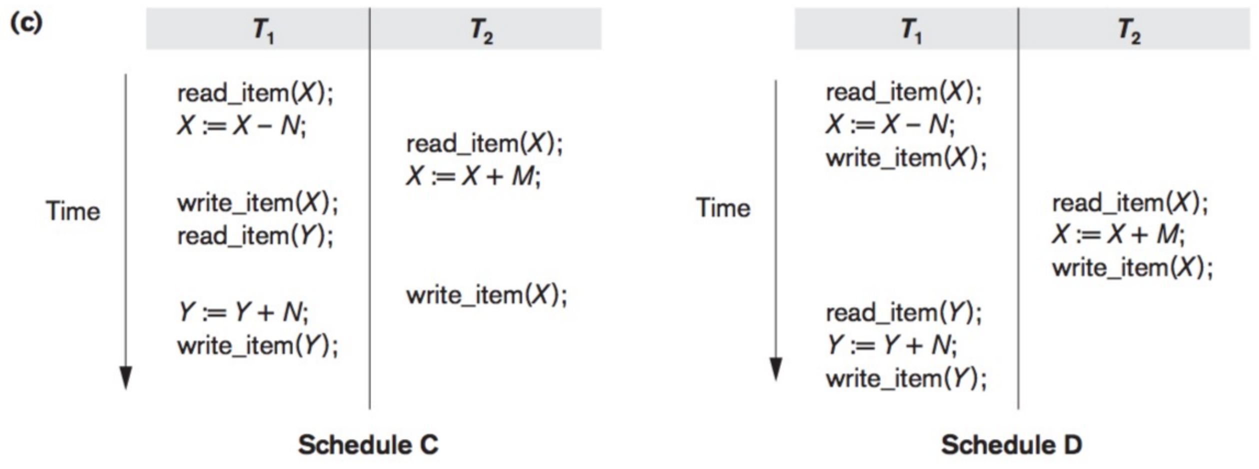
(b)

Time ↓	T_1	T_2
		read_item(X);
		$X := X + M$;
		write_item(X);
	read_item(X);	
	$X := X - N$;	
	write_item(X);	
	read_item(Y);	
	$Y := Y + N$;	
	write_item(Y);	

Schedule B

Serial

What is ACID?



(c)

Time ↓	T_1	T_2
	read_item(X);	
	$X := X - N$;	
		read_item(X);
		$X := X + M$;
	write_item(X);	
	read_item(Y);	
		write_item(X);
	$Y := Y + N$;	
	write_item(Y);	

Schedule C

Time ↓	T_1	T_2
	read_item(X);	
	$X := X - N$;	
	write_item(X);	
		read_item(X);
		$X := X + M$;
		write_item(X);
	read_item(Y);	
	$Y := Y + N$;	
	write_item(Y);	

Schedule D

Non-serial

+ Example: Recovery

T_1	T_2	T_3	T_4
read_item(A)	read_item(B)	read_item(A)	read_item(B)
read_item(D)	write_item(B)	write_item(A)	write_item(B)
write_item(D)	read_item(D)	read_item(C)	read_item(A)
	write_item(D)	write_item(C)	write_item(A)

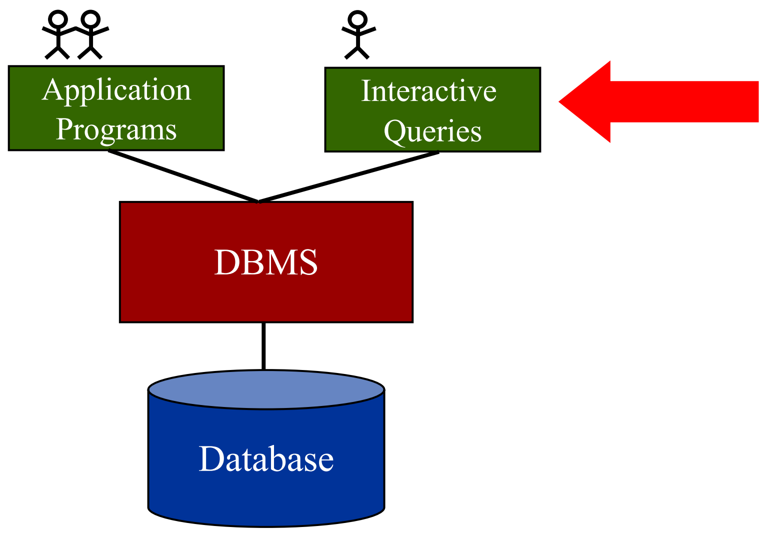
[start_transaction, T_1]

[write_item, T_1, D, 20]
[commit, T_1]
[checkpoint]
[start_transaction, T_4]
[write_item, T_4, B, 15]
[write_item, T_4, A, 20]
[commit, T_4]
[start_transaction, T_2]
[write_item, T_2, B, 12]
[start_transaction, T_3]
[write_item, T_3, A, 30]
[write_item, T_2, D, 25] $\longleftarrow$ System crash

What are undo and redo?

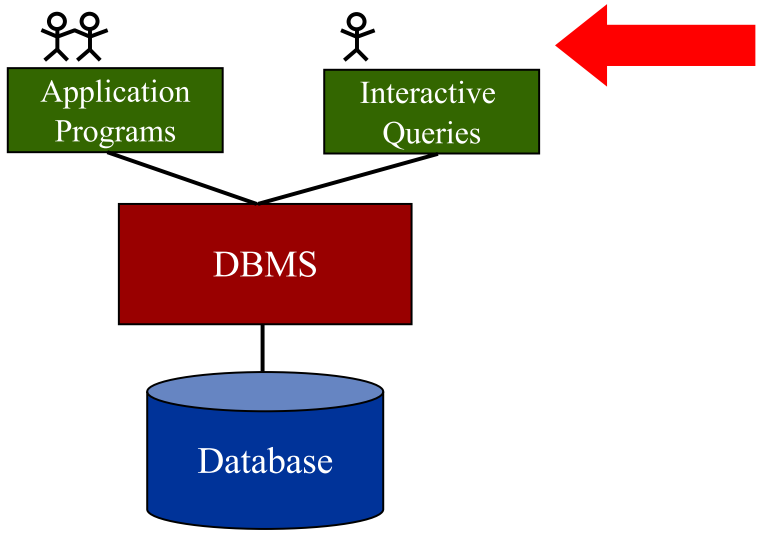
What is checkpoint?

+ The Applications



- Interface
  - View Design
  - Ad hoc querying in SQL
  - Host languages
  - Human Computer Interaction
- Application Programs
  - Functional Analysis
  - Data Flow Diagrams
  - Software Engineering

+ The Users



- Database Administrators
- Database Designers
- End Users
- Application Programmers

+ What is a Relational Query

- Data in a relational database can be manipulated in the following ways:
  - INSERT: New tuples may be inserted
  - DELETE: Existing tuples may be deleted
  - UPDATE: Values of attributes in existing tuples may be changed
  - SELECT: Attributes of specific tuples, entire tuples, or even entire relations may be retrieved
- Relational query languages should provide all the above

+ Relational Query Languages

Relational queries are formulated in relational query languages

- Relational Algebra (RA)
  - Formal query language for a relational database
- Structured Query Language (SQL)
  - Implementation of RA
  - Comprehensive, commercial query language with widely accepted international standard
- Query by Example (QBE)
  - Commercial, graphical query language with minimum syntax

…relational calculus is also part of the relational model for databases and provides a declarative way to specify database queries. The relational algebra is procedural.

+ SQL

- SQL is considered one of the major reasons for the success of relational model in the database industry
- SQL provides an industry wide standard for database access
  - In practice different product support different dialects of SQL
- It is a declarative language for users to specify what the result of the query should be, DBMS decides operations and order of execution
- SQL is designed for data definition, data manipulation, and data control, powerful enough to retrieve any piece of data from database

+ Three Types of SQL Statements

- Data Definition Language (DDL)
  - Statements to define the database schema
- Data Manipulation Language (DML)
  - Statements to manipulate the data
- Data Control Language (DCL)
  - Statements to specify transaction control, semantic integrity (triggers and assertions), authorization and management of privileges
  - Statements for specifying the physical storage parameters such as file structures and access paths (indexes)
  - Statements to specify the role-based security controls

+ DDL - Data Definition Language

- Statements to define the data
  - These statements create (and change) the database schema
- Basic SQL DDL Statements
  - Table Definition
    - CREATE TABLE
    - DROP TABLE
    - ALTER TABLE

+ CREATE TABLE

- It creates a new relation, by specifying its name, attributes and constraints, and records the table definition in the system catalog
  - The key, entity and referential integrity constraints are specified within the statement after the attributes have been declared
  - The domain constraint is specified for each attribute by giving a valid (SQL99) data type and (optionally) excluding NULL from the domain
    - Valid SQL99 data types include INTeger, CHARacter, DATE, DECIMAL, etc.
    - Data type of an attribute can be specified directly or by declaring a domain (CREATE DOMAIN)

+ CREATE TABLE Syntax

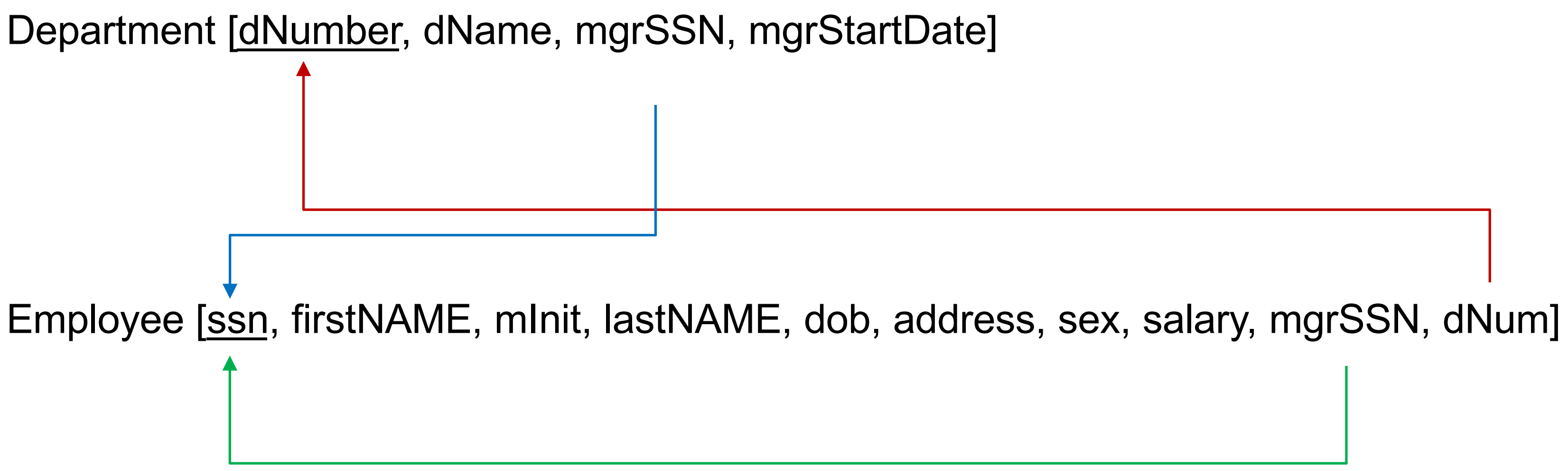
```
CREATE TABLE <table name>
  (<column name> <column type> [<attribute constraint>]
  {, <column name> <column type>  [<attribute constraint>] }
  [<table constraint> {, <table constraint>} ] )
```

Notations:

KEY WORDS, "," "(" ")"
<name>,
{repeat 0-n times},
[optional]

...to learn complete SQL syntax and examples, visit https://www.w3schools.com/sql.
...you should keep a SQL syntax quick reference handy

+ CREATE TABLE Example



Department [dNumber, dName, mgrSSN, mgrStartDate]

Employee [ssn, firstNAME, mInit, lastNAME, dob, address, sex, salary, mgrSSN, dNum]

Partial Relational Schema for Company Database

+ CREATE TABLE Example (1)

```
CREATE TABLE Employee
  (   firstName       VARCHAR (15)        NOT NULL,
      mInit           CHAR,
      lastName        VARCHAR (15)        NOT NULL,
      ssn             CHAR (9)            NOT NULL,
      dob             DATE,
      address         VARCHAR (30),
      sex             CHAR,
      salary          DECIMAL (10, 2),
      mgrSSN          CHAR (9),
      dNum            INT                 NOT NULL,
  PRIMARY KEY (ssn),
  FOREIGN KEY (mgrSSN) REFERENCES Employee (ssn),
  FOREIGN KEY (dNum) REFERENCES Department (dNumber) );
```

+ CREATE TABLE Example (2)

Constraints can be given a name:

```
CREATE TABLE Employee
 (   firstName        VARCHAR (15)          NOT NULL,
     …….
     ssn              CHAR (9)              NOT NULL,
     mgrSSN           CHAR (9),
     dNum             INT                   NOT NULL,
 CONSTRAINT empPK PRIMARY KEY (ssn),
 CONSTRAINT smpMgrFK FOREIGN KEY (mgrSSN)
           REFERENCES Employee (ssn),
 CONSTRAINT empDNumFK FOREIGN KEY (dNum)
           REFERENCES Department (dNumber)
 );
```

+ CREATE TABLE Example (3)

- A referential triggered action clause can be attached to a foreign key constraint, to specify the action to take if a referenced tuple is deleted, or a referenced primary key value is modified

ON DELETE SET NULL | SET DEFAULT | CASCADE

ON UPDATE SET NULL | SET DEFAULT | CASCADE

```
CREATE TABLE Employee
(…….
   dNum           INT      NOT NULL DEFAULT 100,
   …….
   FOREIGN KEY (dNum) REFERENCES Department (dNumber)
          ON DELETE SET DEFAULT
          ON UPDATE CASCADE);
```

+ Enforcing Referential Integrity

- movieID in StarsIn is a foreign key that references Movie
  - StarsIn.movieID → Movie.movieID
- What should be done if a Movie tuple is deleted, and there is a StarsIn tuple refers to it?
  1. Delete all roles that refer to it?
  2. Disallow the deletion of the movie?
  3. Set moveID in StartsIn tuples that refer to it to null?
  4. Set moveID in StartsIn tuples that refer to it to default value?

By default no action is taken and the delete/update is rejected.
Other actions include :

ON DELETE SET NULL | SET DEFAULT | CASCADE
ON UPDATE SET NULL | SET DEFAULT | CASCADE

+ So It Looks Like This

```
CREATE TABLE  StarsIn (
  starID      INTEGER,
  movieID  INTEGER,
  role         CHAR(20),

  PRIMARY KEY (starID, movieID),
  FOREIGN KEY (starID)  REFERENCES MovieStar
      ON DELETE CASCADE
      ON UPDATE CASCADE,
  FOREIGN KEY (movieID)  REFERENCES Movie
       ON DELETE SET NULL
       ON UPDATE CASCADE)
```

+ Question:

Consider the following table definition.

```
CREATE TABLE  ParkingPermit  (
    pID       INTEGER,
    staffID  INTEGER,  …

    PRIMARY KEY (pID),
    FOREIGN KEY (staffID) REFERENCES Staff  ON DELETE CASCADE);
```

Assume there is a tuple with pID = 1000 and staffID = 5678 in the table, choose the best answer

1. If the row for staffID value 5678 in Staff is deleted, then only the row with pID = 1000 in ParkingPermit is automatically deleted
2. If the row with staffID value 5678 in Staff is deleted, then all rows with staffID=5678 in ParkingPermit are automatically deleted
3. Both of the above

Based on the instruction given in the table definition, only option 2 is correct

+ ALTER TABLE

- ALTER TABLE command is used for schema evolution, that is the definition of a table created using the CREATE TABLE command, can be changed using the ALTER TABLE command
- Alter table actions include
  - Adding or dropping a column
  - Changing a column definition
  - Adding or dropping constraints

+ DROP TABLE

- **DROP TABLE**
  - Drops all constraints defined on the table including constraints in other tables which reference this table
  - Deletes all tuples within the table
  - Removes the table definition from the system catalog
- DROP TABLE Syntax

  DROP TABLE <table name> [**CASCADE**];

+ DML - Data Manipulation Language

- Statements to manipulate the data
  - These statements work on database instances
  - Both input and output are tables
- Basic SQL DML Statements
  - **SELECT**
  - **INSERT**
  - **DELETE**
  - **UPDATE**

+ INSERT Statement

INSERT statement is used to add tuples to an existing relation

- Single Tuple INSERT
  - Specify the relation name and a list of values for the tuple
  - Values are listed in the same order as the attributes were specified in the CREATE TABLE command
  - User may specify explicit attribute names that correspond to the values provided in the insert statement. The attributes not included cannot have the NOT NULL constraint
- Multiple Tuple INSERT
  - By separating each tuple’s list of values with commas
  - By loading the result of a query

+ INSERT Syntax

```
INSERT INTO <table name>
    [(<column name> {, <column name> })]
(VALUES (<constant value>, {,<constant value> })
        | <select statement>);
```

…syntax errors can occur

+ INSERT Example: From Values

```
INSERT INTO Employee
VALUES
(‘Richard’, ‘K.’, ‘Marini’, ‘653298653’,
‘30-DEC-1995’, ‘98 Brisbane Street, St Lucia, QLD’,
‘M’,37000, ‘987654321’,4);
```

Employee [firstName, mInit, lastName, ssn, dob, address, sex, salary, mgrSSN, dNum]

...can also insert for some selected columns
...integrity checks will be performed
...values with quotation or not can be flexible (i.e., supporting automatic data type convention)

+ INSERT Example: From Queries

```
INSERT INTO DeptInfo
(dName, numOfEmployees, totalSalary)
        SELECT      dName, COUNT (*), SUM (salary)
        FROM        Department, Employee
        WHERE       dNumber = dNum
        GROUP BY    dName ;
```

Employee [firstName, mInit, lastName, ssn, dob, address, sex, salary, mgrSSN, dNum]
Department [dName, dNumber, mgrSSN, mgrStartDate]

…we will discuss SELECT statement later

+ DELETE Statement

DELETE statement is used to remove existing tuples from a relation

- A single DELETE statement may delete zero, one, several or all tuples from a table
- Tuples are explicitly deleted from a single table
- Deletion may propagate to other tables if referential triggered actions are specified in the referential integrity constraints of the CREATE (ALTER) TABLE statement

…DELETE all tuples doesn't equivalent to DROP a table

+ DELETE Syntax

```
DELETE FROM <table name>
    [WHERE <select condition>];
```

```
DELETE FROM Employee
WHERE dNum = 5;
```

+ UPDATE Statement

UPDATE statement is used to modify attribute values of one or more selected tuples in a relation.

- Tuples are selected for update from a single table
- However, updating a primary key value may propagate to other tables if referential triggered actions are specified in the referential integrity constraints of the CREATE (ALTER) TABLE statement

```
UPDATE Employee
SET salary = salary * 1.1
WHERE lastName = ‘McGowen’;
```

+ SELECT Statement

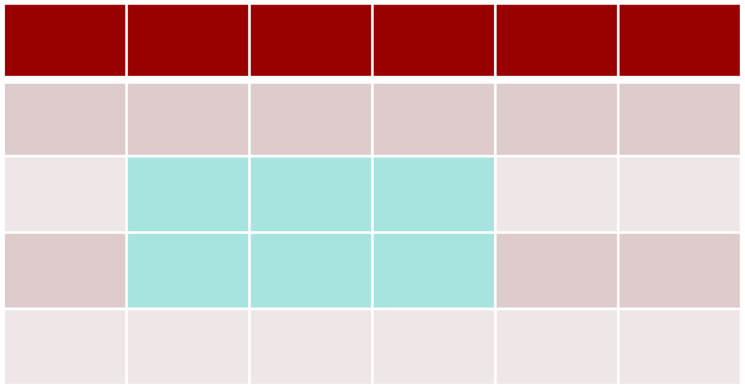
- SQL has one basic statement for retrieving information from the database
- In the SELECT statement, users specify what the result of the query should be, and the DBMS decides the operations and order of execution, thus SQL queries are declarative

+ SELECT Basic Syntax

SELECT <attribute list>

FROM <table list>

[**WHERE** <condition>] ;



- <attribute list> is a list of attribute names (or an expression) whose values are to be retrieved by the query
- <table list> is a list of relation names required to process the query
- <condition> is a conditional (Boolean) expression that identifies the tuples to be retrieved by the query

+ Simple SELECT Example

```
SELECT address
FROM    Employee
WHERE firstName = “Joe” AND lastName = “Bates”;
```

```
SELECT count(*)
FROM    Employee
WHERE mgrSSN = “12345”;
```

Employee [firstName, mInit, lastName, ssn, dob, address, sex, salary, mgrSSN, dNum]

+ Review

- Do you know …
  - What is a Relational Query
  - What are the three main classes of SQL statements
  - What is the syntax for each of the three classes of SQL statements?
- Next …
  - SQL is a very powerfull language, but it is not hard to under the basics of SQL – if you are not familiar with SQL, you read a database textbook or an online tutorail sbout SQL

The content in this review is not part of assessment.

+ Data Storage

- Hierarchical storage structure
  - Primary
    - RAM (random access memory)
    - Fast: access in nanoseconds (10^{-9} second)
    - Relatively small, expensive and volatile
  - Secondary
    - Disk
    - Transfer to RAM in microseconds (10^{-6} second)
    - Locate data in milliseconds (10^{-3} second)
    - Large, non-volatile but slow
  - Tertiary
    - Tape

…how to retrieve data from disks
… trends in pricing, speed and database sizes

+ Data Retrieval

- Data in a system is referenced by address
- SQL queries reference data by content
  - The values of attributes
- Thus, a search is necessary to find the data of given conditions
- Linear search is very expensive
  - Assume 10^5 records/second
  - For a bank with 10^7 accounts, and there is one transaction per account per day
  - Time: $10^7 * 10^7 / 10^5 = 10^9$ seconds = 11,574 days = 31 years!

+ Files of Records

- File: a collection of pages (aka blocks), each containing a collection of records.
  - insert/delete/modify records
  - read a particular record (specified using record id)
  - scan all records (possibly with some conditions on the records to be retrieved)
- Performance can be improved by using better devices
  - Larger main memory and non-volatile memory
  - Faster/parallel disks
  - Buffers: reduce the number of disk access

+ Smarter File Structures

- Heap file (i.e., unordered records)
  - Fast write, but a linear scan is required for search
- Ordered records
  - Sorted by a key field
  - Fast search (e.g., binary search), but insert is slower
- Hashing techniques
  - Hashed by a key field
  - The address of a key field value can be calculated
  - Superfast for equality-based search
  - Problems: overflow/collision and non-support for other types of search (inequality-based, range queries)

+ Indexes

- An index is a data structure that allows direct access to a row in a table (i.e., a record in a database)
  - That is, to find the records satisfying a given condition
- There are different types of indexes:
  - Primary indexing: defined mainly on the primary key of the data-file, where the data-file is already ordered based on the primary key values
  - Clustered indexing: the data sharing *similarity* (i.e., likely to be accessed together) are stored physically together
  - Secondary indexing: an auxiliary data structure that helps to locate the records
    - The actual data is not organised, but we have an ordered reference to where the data points are

+ Some Comments on Indexes

- A table can have many indexes
- An index can be defined on a combination of attributes
- An index can still be very large, therefore multi-level indexes or tree-structures can be used
- It can be costly to create and maintain an index

+ SQL for Creating Indexes

- A DBA or an application programmer is responsible for creating indexes
  - Which tables? which attributes? which index types?
  - Need to create and drop according to application patterns
- SQL DDL statements

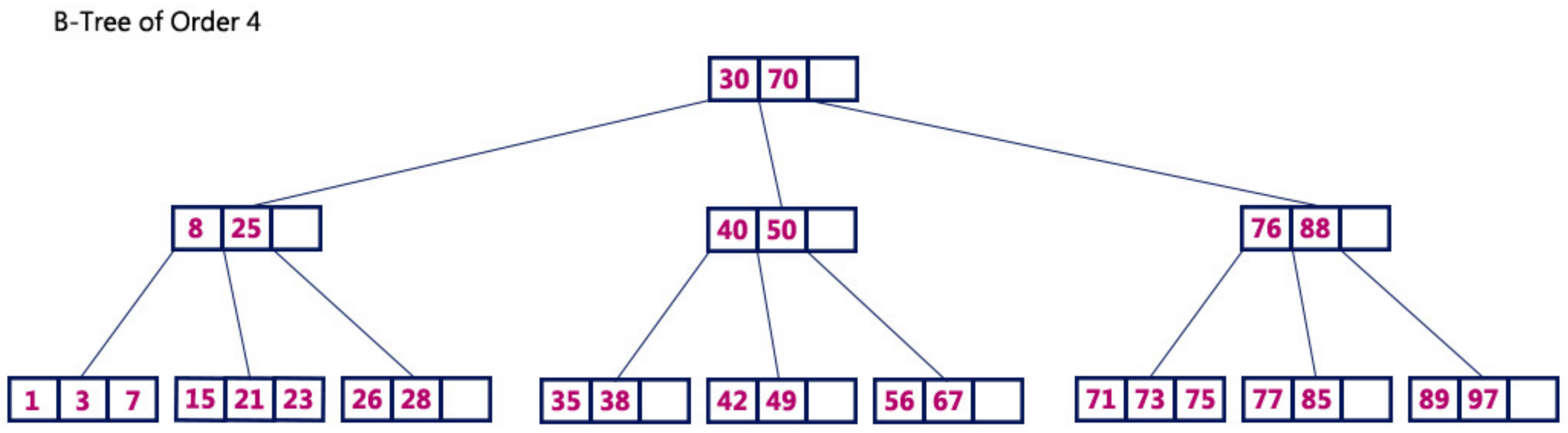
```
CREATE [ UNIQUE | BITMAP ] INDEX index_name
ON table_name (column [ASC | DESC] [,…]);


DROP INDEX index_name;
```

+ Three Main Index Types in RDBMS

- B-tree and B+-tree indexing
- Bitmap indexing
- Hashing indexing

+ An Example of B-Tree



- A balanced tree (insert/delete/update can cause significant tree restructuring)

+ B$^+$-Tree

- In a B-Tree, every value of the search field appears once at some level in the tree
  - In a B$^+$-Tree, data pointers are stored only at the leaf nodes of the tree
  - The leaf nodes have an entry for every value of the search field, along with a data pointer to the record
- Most implementations of a dynamic multi-level index use B$^+$-Tree
  - Leaf nodes for data
  - Internal nodes for search
  - Leave nodes are linked together (for ordered access)

+ Searching Using B$^+$-tree

- Given a value *k* and a B$^+$-tree index *b*
- Compare *k* with the key field values in *b*, to find a sub-tree which might contain *k*
  - Let *b* be the sub-tree found
  - If *b* is already a leaf node, *k* should be a pointer to the block where *k* is stored, if exist
    - Report the record found, or report non-existence
  - If *b* is not a leaf node, repeat this step

… the maximum number of recursion is the height of the tree
… O(n) vs O(log n)

+ Updating a B$^+$-tree

- Insert
  - Search to locate where to insert
  - When a non-root node is full and a new entry is inserted into it, that node is split into two at the same level, and the middle entry is moved up
  - Splitting can propagate all the way to the root node, creating a new level if the root is split
- Delete
  - Search to locate where to delete
  - If deletion of a value causes a node to be less than half full, it is combined with its neighboring nodes, and this can also propagate all the way to the root
- Update = Delete + Insert

+ Bit-Map Indexing

- An array of bits
  - One bitmap for each distinct value of the attribute
  - One bit map has as many bits as the number of tuples
  - The i^{th} bit is 1 if the i^{th} tuple has the value, 0 otherwise
- Suitable for
  - When a table has million of records and columns have low cardinality
  - When queries often use a combination of multiple WHERE conditions involving the logic operators (AND , OR etc)
  - When there is ready-only or low update activity on the key columns

+ An Example for Using Bitmap Index

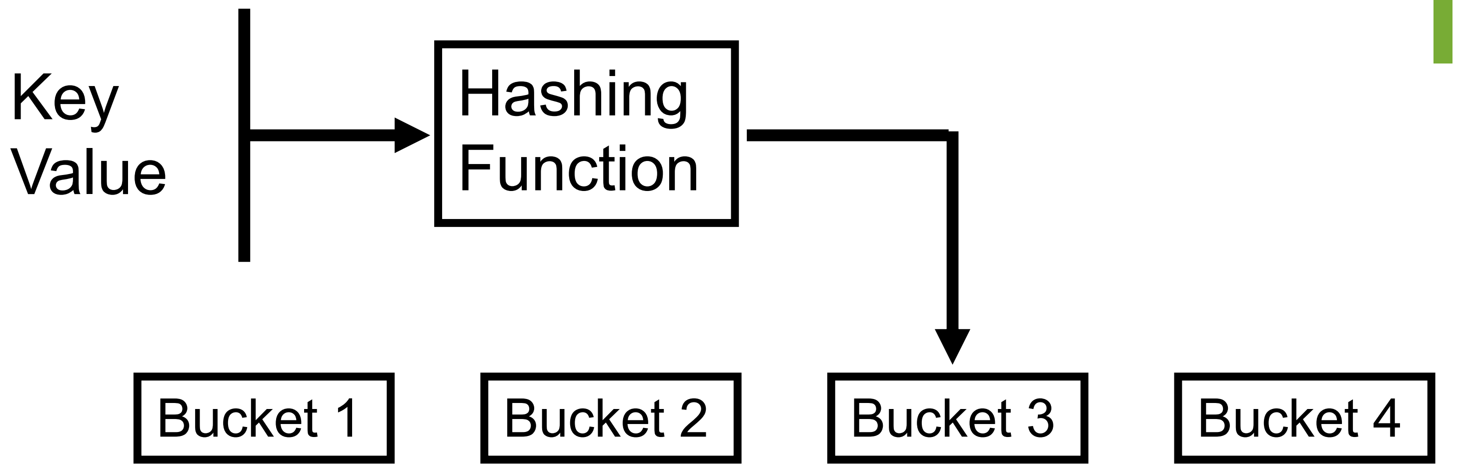
```
SELECT      *
FROM        emp
WHERE       dept = ‘A’ AND
            job = ‘Sales’ AND
            gdr = ‘F’;
```

Bitwise AND:

dept:	111000
sales:	101000
gdr:	011101
emp:	001000

	EMP Table					dept				job				gdr	
	ID	**DEPT**	**Job**	**GDR**		A	B	C		Sales	R&D	Admin		M	F
0	10	A	Sales	M		1	0	0		1	0	0		1	0
0	20	A	R&D	F		1	0	0		0	1	0		0	1
1	30	A	Sales	F		1	0	0		1	0	0		0	1
0	40	B	R&D	F		0	1	0		0	1	0		0	1
0	50	B	Admin	M		0	1	0		0	0	1		1	0
0	60	C	R&D	F		0	0	1		0	1	0		0	1

+ Hashing Indexing

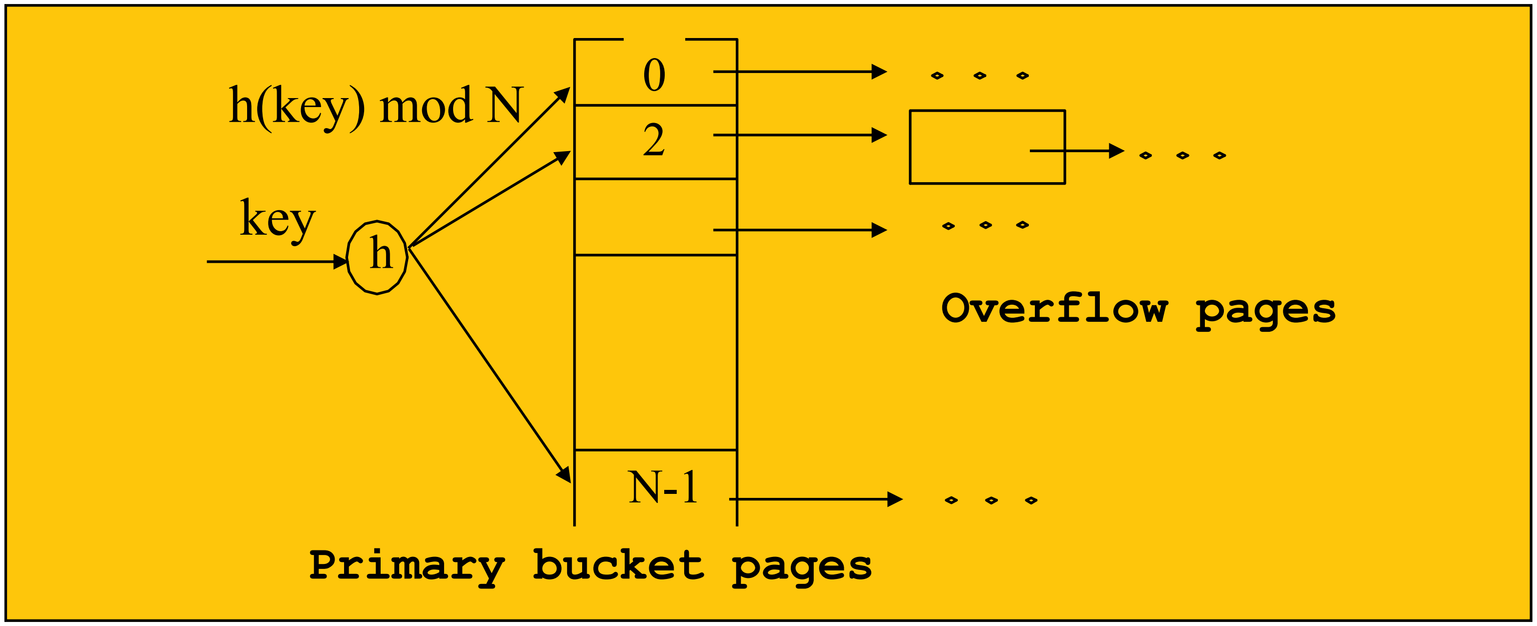


- KV-to-address mapping is computed, not by search
- Ideal for equality queries
- Easy-to-implement, simple syntax
- Not ideal for non-equality search and range queries

+ Hashing Index Design

- A hash function f: mapping attributes to bucket numbers
  - Many-to-1 mapping (why not 1-to-many or many-to-many mapping?)
  - Goal: Spread data evenly over the buckets (why?)
  - Questions:
    - How many buckets? Do you know this beforehand?
    - How big should they be?
    - What happens when a bucket is full?
- Search and insert with a hashing index
  - The same hash function is used for data distribution and search
  - Overflow buckets will need to be used
    - Dynamic hashing can grow or shrink the number of buckets as necessary

+ An Illustrative Example



+ Variations of Hashing Indexing

- Static Hashing
- Linear hashing
- Extendible hashing

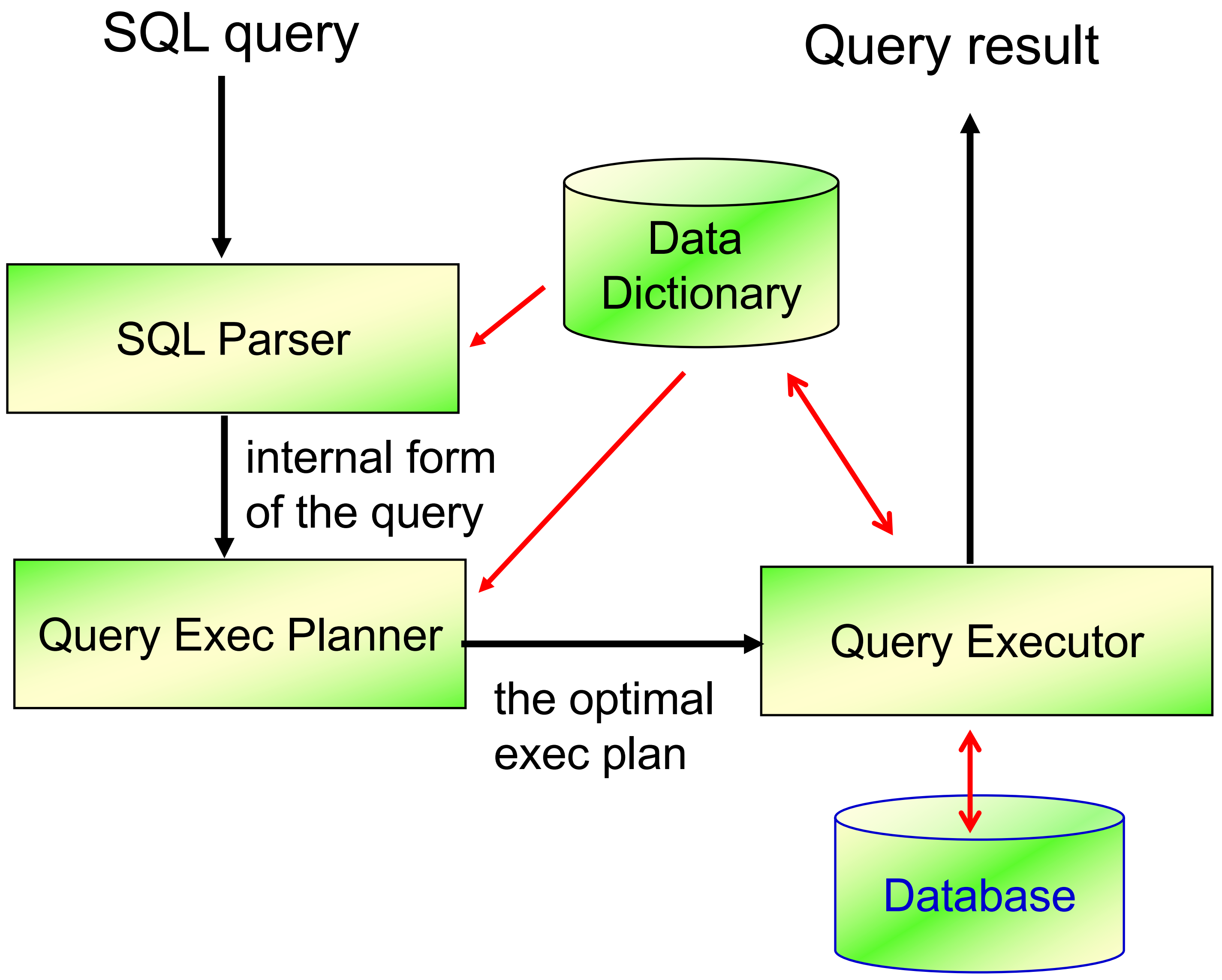
… we will come back to these later when we introducing spatial indexing methods

+ Questions

- Can you compare these three types of indexing methods
  - How to create such indexing?
  - How to use each type of index for your query processing?
  - What's the pros and cons of each type of indexes?
  - Are they easy to be maintained (i.e., how good are they to support insert/delete/update)?
- Can they support multi-dimensional data?

Three types of indexing methods: B-tree/B$^+$-tree, bitmap and hashing

+ Query Processing



+ Relational Algebra

student(sID, program, other)

course(cID, title)

enrolment(sID, cID, grade)

```
SELECT course.title, student.program
FROM   student, course, enrolment
WHERE  enrolment.sID = student.sID AND
       enrolment.cID = course.cID AND
       enrolment.grade = ‘A’;
```

Query:

Find the course titles and the programs that students achieved grade A.

$\pi_{\text{title, program}}(\sigma_{\text{grade='A'}}(\text{student} \bowtie \text{enrolment} \bowtie \text{course}))$

… a relational algebra query implies an order of execution, but it can be re-written without changing its semantic meaning

+ Equivalent RA Expressions

$\pi_{\text{title, program}}(\sigma_{\text{grade='A'} \wedge \text{enrolment.sID = student.sID} \wedge \text{enrolment.cID = course.cID}}(\text{student x enrolment x course}))$

as

$\pi_{\text{title, program}}(\sigma_{\text{grade='A'}}(\text{course} \bowtie (\text{enrolment} \bowtie \text{student})))$

or

$\pi_{\text{title, program}}(\pi_{\text{title, sID}}(\text{course} \bowtie \sigma_{\text{grade='A'}}\text{ enrolment}) \bowtie \text{student})$

or

$\pi_{\text{title, program}}(\pi_{\text{cID, program}}(\text{student} \bowtie \sigma_{\text{grade='A'}}\text{ enrolment}) \bowtie \text{course})$

… there is a theoretical foundation about equivalent rewritings

+ Transformation Rules

- Possible only because the relational algebra is a branch of mathematics, and has rewrite rules giving provably equivalent forms
- Join commutes $A \bowtie B = B \bowtie A$
- Join is associative $(A \bowtie B) \bowtie C = A \bowtie (B \bowtie C)$
- Selection distributes over join
  $\sigma_P(A \bowtie B) = \sigma_{P1} A \bowtie \sigma_{P2} B$
- Projection can be copied through joins
  $\pi_C(A \bowtie B) = \pi_C(\pi_{CX} A \bowtie \pi_{CX} B)$
- and several others

+ Query Simplification Rules

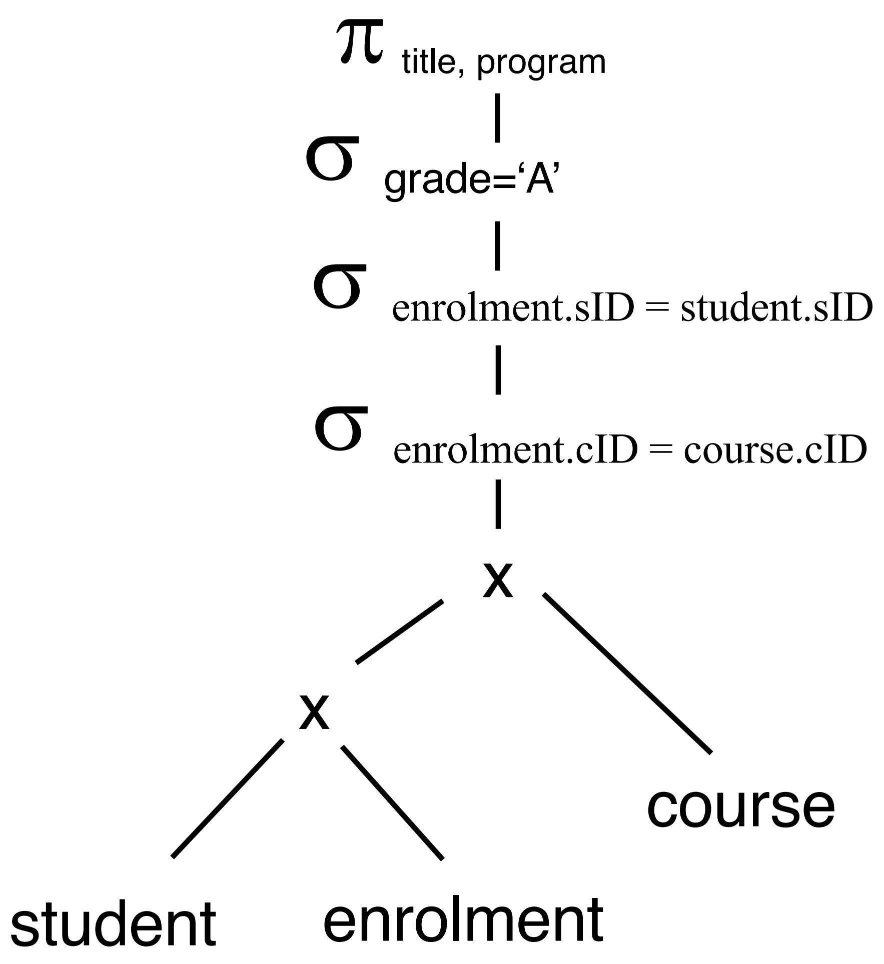
- $R \cup R \equiv R$
- $R \cap R \equiv R$
- $R \bowtie R \equiv R$
- $R - R \equiv \emptyset$
- $R \cup \emptyset \equiv R$
- $R \cap \emptyset \equiv \emptyset$
- $R \bowtie \emptyset \equiv \emptyset$
- $R - \emptyset \equiv R$
- $\emptyset - R \equiv \emptyset$
- $\pi_x \emptyset \equiv \emptyset$

+ Query Execution Trees

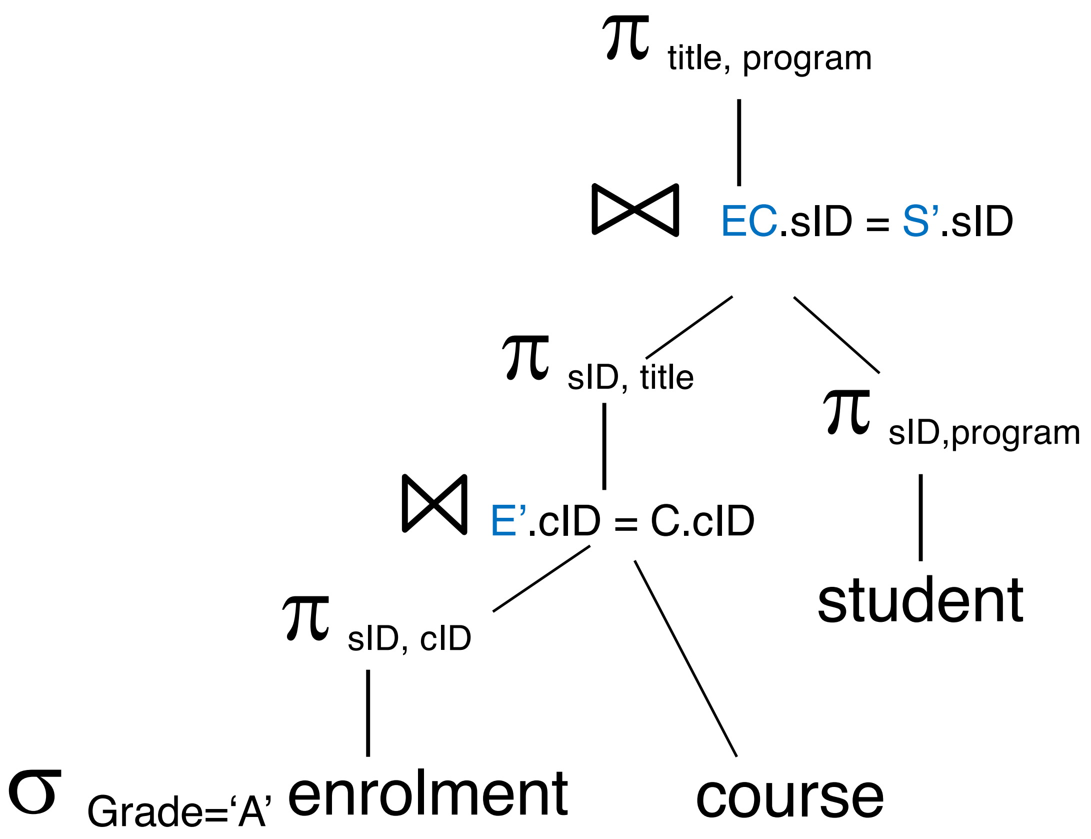
- Query tree: a data structure that corresponds to a relational algebra expression
  - Input relations of the query as leaf nodes
  - Relational algebra operations as internal nodes
- An execution of the query tree consists of executing internal node operations

+ An Initial Execution Tree

$\pi_{\text{title, program}}(\sigma_{\text{grade='A'} \wedge \text{enrolment.sID = student.sID} \wedge \text{enrolment.cID = course.cID}}$ (student x enrolment x course))



+ A Better Execution Tree



+ Query Execution Optimisation

- Map an SQL query to an execution tree
  - Use multi-way Cartesian products to combine tables
- Push down selections
  - What can be pushed down?
  - Avoiding Cartesian products, and use only binary joins
- Push down projections
  - What can be pushed down?
- Simplify using constraints and constants
- Apply other optimisation strategies
  - Common sub-expressions, for example

… minimize intermediate table sizes

…often require database statistics to make a good estimation

+ Summary

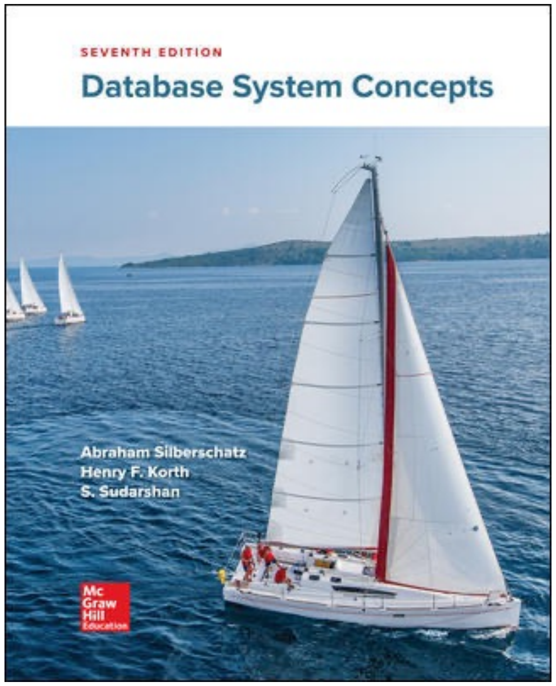
- We have reviewed RDBMS (typically covered in 1~2 courses)
- We paid particular attention to
  - Database concepts
  - Creating tables (data types and constraints)
  - SQL (syntax, use of attributes and constants with data types)
  - Indexing (B-trees, bitmaps and hashing indexing)
  - Query optimisation (equivalent forms and execution trees)
- These concepts and techniques will be revisited for spatial and multimedia databases
  - Many things are common, at least conceptually

+ Readings

Textbook (any good ones)

Database System Concepts, 7th Edition

A. Silberschatz, H.F. Korth, and S. Sudarshan, McGraw-Hill, 2020 (OK to use any edition)



Focusing on

- SQL languages
- Indexing
- Query execution planning and optimization