# ER Diagram

### Many to Many

A can have multiple B and B can have multiple A



### One to Many

A can have multiple B but B can only have one A



### One to One

A can have only one B and B can have only one B



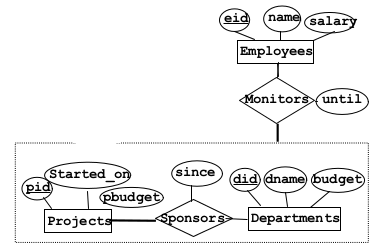
### At least one

Bold arrow specify there is at least one element.

|  |  |
| --- | --- |
|  | A have at least one |
|  | A have exactly one |

### Aggregation

Allows us to treat a reltionship set R as an entity set so that R can participate in other relationships



# Relational algebra

|  |  |
| --- | --- |
| Selection |  |
| Projection |  |
| Renaming |  |
| Cross product |  |
| Join |  |
| Division |  |
| Intersection |  |
| Union |  |
| Set different |  |

## Type of joins

|  |  |
| --- | --- |
| Real life | SQL |
| http://howtorollmarijuana.com/wp-content/uploads/2013/07/joints.gif | http://www.codeproject.com/KB/database/Visual_SQL_Joins/Visual_SQL_JOINS_orig.jpg |

* Condition/Theta Join
* Equi Join: in2 Condition join where condition contains ONLY equalities
* Natural Join: Equijoin on all common attribute

# Sql

## Datatype

|  |  |
| --- | --- |
| Char(n) | A character string of fixed length n |
| VarChar(n) | Denotes a string of up to n charaters |
| INT or INTEGER | An integer |
| SHORTINT | Smaller integer |
| FLOAT or REAL | Float number |
| DOUBLE PRECISION | Double |
| DATE | Date format YYYY-MM-DD |
| TIME | Time format: hh:mm:ss |

## Table operations

|  |
| --- |
| --Create table  CREATE TABLE Students  (  id INT NOT NULL,  name VARCHAR(20),  login CHAR(10),  major VARCHAR(20) DEFAULT 'undefined',  school\_id INT,  PRIMARY KEY(id),  FOREIGN KEY(school\_id) REFERENCES School(id)  )  --Drop table  DROP TABLE Students  --Alter table  ALTER TABLE Students ADD COLUMN firstyear:integer |

## Row operation

|  |
| --- |
| --INSERT  INSERT INTO Students (id, name, faculty) VALUES (8908998, 'Dupont', 'Science')  --Delete  DELETE FROM Students WHERE id = 0894984  --Update  UPDATE Students SET faculty = 'Arts' WHERE id = 9849849 |

# XML

WHAT DAFUQ?!!@#!@#!?

## DTD

|  |  |
| --- | --- |
| <!DOCTYPE DiscoverTheWorld [  <!ELEMENT DiscoverTheWorld (tour\*,reservation\*)>  <!ELEMENT tour (type, start-date, duration, price) >  <!ELEMENT reservation (cname, caddress, cost, special\*)>  <!ATTLIST tour TourId **ID** **#REQUIRED** >  <!ATTLIST reservation ResID **ID** **#REQUIRED**  TourID **IDREF** **#REQUIRED**>  <!ELEMENT type (**#PCDATA**) >  <!ELEMENT start-date (**#PCDATA**) >  <!ELEMENT duration (**#PCDATA**) >  <!ELEMENT price (**#PCDATA**) >  <!ELEMENT cname (**#PCDATA**) >  <!ELEMENT caddress (**#PCDATA**) >  <!ELEMENT cost (**#PCDATA**) >  <!ELEMENT special (**#PCDATA**) >  <!ATTLIST special price **CDATA** **#REQUIRED**>  ]> | <!DOCTYPE Politics [ <!ELEMENT Politics (Politician\*, Province\*)>  <!ELEMENT Politician ((CurrentMayor | CurrentMop)?, address?)>  <!ELEMENT CurrentMayor (since?)>  <!ELEMENT CurrentMoP (since?)>  <!ELEMENT Province (City+, Riding+, population?)>  <!ELEMENT City (population?)>  <!ELEMENT Riding (population?)>  <!ELEMENT address (**#PCDATA**) >  <!ELEMENT since (**#PCDATA**) >  <!ELEMENT population (**#PCDATA**) >  <!ATTLIST Politician pname **ID** REQUIRED  website **CDATA** IMPLIED  friends **IDREFS** IMPLIED>  <!ATTLIST CurrentMayor cityID **IDREF** REQUIRED>  <!ATTLIST CurrentMoP rname **IDREF** REQUIRED>  <!ATTLIST Province pname **ID** REQUIRED>  <!ATTLIST City cityID **ID** REQUIRED  cname **CDATA** REQUIRED>  <!ATTLIST Riding rname **CDATA** REQUIRED>  ]> |
| **<bibliography>**  **<books>**  **<book** ISBN="23456" year="1995"**>**  **<title>** Foundations ... **</title>**  **<author>** Hull **</author>**  **<author>** Abiteboul **</author>**  **<publ>** Addison Wesley **</publ>**  **</book>**  **<book>** ...**</book>**  **</books>**  **<journals>**  **<journal>**  **<title>** ... **</title>**  **<article>** ... **</article>**  ...  **</journal>**  **<journal>** ... **</journal>**  **</journal>**  **</bibliography>** | **<DiscoverTheWorld>**  **<tour** TourId="1"**>**  **<type>** Brazil junge **</type>**  **<start-date>** 16-April **</start-date>**  **<duration>** 14 **</duration>**  **<price>** 2229 **</price>**  **</tour>**  **<tour** TourId="2"**>**  **<type>** Brazil junge **</type>**  **<start-date>** 30-April **</start-date>**  **<duration>** 21 **</duration>**  **<price>** 2999 **</price>**  **</tour>**  **<tour** TourId="3"**>**  **<type>** Kenia safari **</type>**  **<start-date>** 30-April **</start-date>**  **<duration>** 21 **</duration>**  **<price>** 3229 **</price>**  **</tour>**  **<reservation** ResId="541" TourId="1"**>**  **<cname>** Bettina Kemme **</cname>**  **<caddress>** Montreal **</caddress>**  **<cost>** 2579 **</cost>**  **<special** price="5"**>** vegetarian **</special>**  **<special** price="20"**>** single **</special>**  **</reservation>**  **<reservation** ResId="542" TourId="2"**>**  **<cname>** Your Name **</cname>**  **<caddress>** Your Address **</caddress>**  **<cost>** 3105 **</cost>**  **<special** price="5"**>** vegetarian **</special>**  **</reservation>**  **</DiscoverTheWorld>** |

|  |
| --- |
| <!DOCTYPE people[  <!ELEMENT people(person\*)>  <!ELEEMNT person(name\*, (lastname|familyname)?)>  <!ATTLIST person PID ID #REQUIRED  age CDATA #IMPLIED  children IDREFS #IMPLIED  mother IDREF #IMPLIED  >  <!ELEMENT name(#PCDATA)>  <!ELEMENT lastname(#PCDATA)>  <!ELEMENT familyname (#PCDATA)>  ]> |

Data types: PCDATA (parsed character data) or CDATA (unparsed)

Attributes

* ID unique identifier (similar to primary key)
* IDREF: reference to single ID
* IDREFS: space-seperated list of references

Values

* can give a default value
* #REQUIRED must exist
* #IMPLIED optional

Specified in an XML file with <!DOCTYPE name SYSTEM "path/to/thing.dtd">

Can use regex style things too. \* is 0 or more. + is 1 or more, (a | b)? is one or the other

## XPATH

## /bibliography/book/author all author elements by root navigating through those elements

## /bibliography/book/@ISBN All ISBN attributes

## //title all title elements anywhere in the document

## /bibliography/\*/title titles of bibliography entries assuming that there could be books, journals, reports, etc...

## /bibliography/book[@year>1995] returns books where the year > 1995

## /bibliography/book[author='FooBar']/@Year returns the years of books written by FooBar

## /bibliography/book[count(author) <2]

* /bibliography/book/author[position()=1]/name position is the location of the node in the node set

## XQuery

|  |  |
| --- | --- |
| For | Let |
| for $b in document("bib.xml")/bib/book  return <result> $b</result> | let $b in document("bib.xml")/bib/book  return <result> $b</result> |
| <result><book>…</book></result>  <result><book>…</book></result>  <result><book>…</book></result>  …  <result><book>…</book></result> | <result>  <book>…</book>  <book>…</book>  ..  <book>…</book>  </result> |

|  |  |
| --- | --- |
| -Basic Queries: **'/'**  to navigate one path at a time  Example:/Bookstore/Book/Title  **'//'** all paths following this  Example://Title  When wanting to access an attribute of an element use @  Example:/Bookstore/Book/data(@ISBN)  **'|'** **OR** operator ONLY USED INSIDE CONDITIONS  Example:/Bookstore/Book|Magazine/Title  **'='** can act like == like in Self-Join Queries below  Navigation accesses:  Example: parent::\* \*::child following-silbling::\*  **-Queries involving CONDITIONS**  **1condition:**  Example:/Bookstore/Book[@Price<30]  Example:/Bookstore/Book/Authors/Author[2] the 2nd author of each element  **2conditions:**  To write a condition, it needs to be inside '[...]' then followed by the output that we are looking for  Example:/Bookstore/Book[@Price<30]/Title  **Condition to find elements that contain other elements:**  Example:/Bookstore/Book[Remark]/Title  **Conditions && Conditions + Some output:**  Example: Looking for a title that has one last name =Ullman and price<90  /Bookstore/Book[@Price<90 and Authors/Author/Last\_Name="Ullman"]/Title  **Conditions Inside Conditions + Some output:**  Example:Looking for a title with author ="Jeffrey Ullman" and price<90  /Bookstore/Book[@Price<90 and Authors/Author/[Last\_Name="Ullman" and First\_Name="Jeffrey"]]/Title  **Conditions && !Conditions + Some output:**  Example:Looking for a title with author ="Ullman" and NOT author="Widom"  /Bookstore/Book[/Authors/Author/Last\_Name="Ullman" and count(/Authors/Last\_Name="Widom"=0]/Title  **The condtion 'contains':**  /Bookstore/Book[contains(Remark, "great")]/Title | **Self-Join Query**  Quering two instances of the database at one and joining them together.Trying to find the magazines wheres theres a book with the same title. Example: doc("BookstoreQ.xml")/Bookstore/Magazine[Title=doc("BookstoreQ.xml")/Bookstore/Book/Title]  **Navigation Accesses**  The name() function returns the name of a tag or element  To find all elements whose parent is not "Bookstore" or "Book"  **/Bookstore/\*[name(parent::\*)!="Bookstore" and name(parent::\*)!="Book"]** |

## XML in DB@^%$^$#

|  |
| --- |
| INSERT INTO MyXML(id, INFO) VALUES (1000,  '<customerinfo cid="1000">  <name>Kathy Jones</name>  <addr country =Canada">  <street>123 fake</street>  <city>Ottawa</city>  <prov-state>Ontario</prov-state>  <pcode-zip>H0H 0H0</pcode-zip>  </addr>  </customerinfo>') |

# Buffer

|  |  |  |
| --- | --- | --- |
| DBMS stores information persistently on (“hard”) disks.  ❑ Unit of transfer main-memory/disk: disk blocks or pages.  ❑ Timing:  ✩ 2- 20 msec for random data block (bad seek time)  ✩ If blocks are sequentially on disk, only +1ms per block  ✩ Compare main memory access: in nanoseconds  ❑ Basic operations (READ/WRITE from/to disk)  ❑ Why disks?  ✩ Cheaper than Main Memory  ✩ Higher Capacity  ✩ Main Memory is volatile | When loading a page from disk:  ✩ Replacement frame must have “pin counter” of 0  ❑ When requesting a page that is in the buffer  ✩ Increment pin counter  ❑ After operation has finished  ✩ Decrement pin counter  ✩ Set dirty bit if page has been modified:  ❑ Frame is chosen for replacement by a replacement policy:  ✩ Only unpinned page can be chosen (pin count = 0)  ✩ Least-recently-used (LRU), Clock, MRU etc. | If requested is not in pool:  ✩ If there is an empty frame, use it  ✩ Else choose an empty frame for replacement. If the frame is dirty (page was modified), write it to disk  ✩ Read requested page into chosen frame  Buffer management in DBMS requires ability to:  ✩ **pin a page** in buffer pool, **force a page to disk** (important for  implementing CC & recovery),  ✩ adjust **replacement policy**, and **pre-fetch pages** based on  access patterns in typical DB operations. |

# Indexing

|  |  |  |
| --- | --- | --- |
| **COST MODEL**  Measure performances by simplifying the parameters (IO focused):  ✩ only consider disk reads (ignore writes)  ✩ only consider number of I/Os and not the individual time for each read (ignores page pre-fetch)  ✩ Average-case analysis; based on several simplistic assumptions. ● delete/update ▲ depends on where | **HEAP FILES**  ✩ Linked, unordered list of all pages of the file  ✩ Is it good for:  ● scan retrieving all records (SELECT \*)?  ▲ yes, you have to retrieve all pages anyway  ● equality search on primary key  ▲ not great: have to read on avg half the pages for 1 record  ● range search or equality search on non-primary key  ▲ not great, all pages need to be read  ● insert  ▲ yes, can insert anywhere  ● delete/update  ▲ depends on where | **SORTED FILES**  ✩ Records are ordered according to one or more attributes of the relation  ✩ Is it good for:  ● scan retrieving all records (SELECT \*)?  ▲ yes, you have to retrieve all pages anyway  ● equality search on sort attribute  ▲ good: find first qualifying page with binary search (log2)  ● range search on sort attribute  ▲ good: find first qualifying page with binary search (log2):  adjacent pages might have additional matching records |

Let suppose we have a relation R (A, B, C, D, F) such that:

* A and B are int (6 byte)
* C-F are char [40] (10 byte per char).

Tuple = 172 bytes. 200,000 tuples

Each data page has 4000 bytes and is around 80% full.

.

B values are uniformly distributed

Non-clustered index B-tree with <k, list of rid>

Index pages are 4K and between 50%-100% fulll

rid = 10 bytes

size of pointer in intermediate page = 8 bytes

Index entry size in root/intermediate pages = size(key) + size(pointer) 6 + 8 = 14 bytes

Average number of rids per data entry = number of uples / different values (if uniform). eg: 200,000/20,000 = 10

Average length per data entry = size of key + (number of rids \* size of rid). eg: 6 + 10\*10 = 106

Average number of date entrier per page = fill-rate \* page size / length of entry. eg 0.75\*5000/106 = 28 entries per page

Estimate number of leaf page = number of different values / number of entrier per page. eg: 20,000 / 28 = 715

Number of entries in intermediate pages = fill-rate \* page size / lneght of index enty. eg: 0.5 \* 4000 / 15 = 143, 1\*4000/14 = 285

Height of tree = Number of leaf pages / (min | max)? number of entries in intermediate pages

Give the pids of all projects within department D2 that started in 2014.

Give the pids of all projects that have at least one excellent evaluation