Relational calculus review

Forall -- universal quantifier
Exists -- existential quantifier

(forall v) condition = not (exists v) not(condition)

****

(forall s) (student(s) and (gpa > 2.0))
= not(exists s) (students(s) and not (gpa > 2.0))
= not(exists s) (students(s) and (gpa <= 2.0))

****

SQL:

```
Select max(gpa) as Max_gpa, avg(gpa) as Avg_gpa, count(*) as students
From Student
Where gpa > 3.0
Group by county
Having count(*) >= 10
Order by students DESC, Avg_gpa DESC;
```

View

- Control access to the data set
  (Limit access to users to a subset of data)
- Provide only the relevant info (avoids confusion with large data set)
- Derive info from DB and present as data to user.
- Rename columns/tables according to user's familiarity

Update through a view is infeasible in many cases.
It is not recommended to support it.

A view is a definition of a query
where the query generates the subset of data from the DB.

View/definition name

```
Create view CustRep
AS
    SELECT c.CustomerName as CustName,
           r.Firstname + r.Lastname as RepName
    From Customer c, Rep r
    Where c.repNum = r.repNum
```

Query

Query against view (by using the view as a table)

```
SELECT CustName
FROM CustRep        -- will be replaced by
Where RepName = '----'
```

```
Drop View CustRep
```

Index

Index expedites retrieval operations.

It slows down insert/update/delete operations.

The index values must be stored in sorted order.

When ~~index~~ a table is updated,
the index entries record no. need to be modified

Index is stored outside the table.

Requires additional storage space.
additional processing for updating index entries for ~~any~~ changes to the table.

Create index indexName
on Tablename(ColumnName);

Drop index indexname on Tablename(ColumnName)

For massive updates,
~~delete~~ drop index
- Perform updates
- Rebuild index

Justification for an index } = frequency of retrieval on / frequency of update on } index Column

Constraints

Rule/condition that must be satisfied by the DB.

1) Entity integrity constraint:

Each entity must be uniquely identifiable in the DB.

Each column of the primary key cannot have null value

2) Referential integrity constraint

Any reference made from a table (through columns) that reference must exist as a valid entity in some table (as the primary key).

This is implemented through Foreign key specification

3) Legal (Domain) value

Permissible set of values for a column (domain).

For the implementation of business specific rules/conditions that cannot be represented with the above constraints, Triggers are used.

change in salary value of an employee cannot be negative.

If (newSalary − oldSalary) < 0 Then

[Prevent this update operation]

Event → Action

Trigger

DBMS maintains two tables

	"INSERTED"	"DELETED"
insertion:	✓	—
deletion:	—	✓
update	new record	old record