Database.

1) Google holds petabytes (10^{15} bytes). This is not held in a traditional DBMS, but in specialized structures optimized for search-engine queries.

2) Satellites send down petabytes of information for storages in specialized systems.

3. A picture is actually worth way more than a thousand words. → use much more space to store image.

4m|c5;

* Query? The query is parsed and optimized by a query compiler. The resulting query plan, or sequence of actions of the DBMS will perform to answer the query, is passed to the execution engine.
→ issues the sequence of request for a small pieces of data, include data file, index file which help find elements of data files queky from (based on the resoura manager).

After that, the request data will passed to the buffer manager. It will bring appropriate portions of the data from storage (disk) to the main-memory buffers.

The Buffer manager communicates with a storage manager to get data from disk.
→ the DBMS issues commands directly to the disk controller.

1. Data: the contents of the database itself.
2. Metadata: the database schema that describes the structure of, and constraints on, the database.
3. Log records: Information about recent changes to the database; these support durability of the database.
4. Statistics: Information gathered and stored by the DBMS about data properties such as the sizes of, and values in, various relations or other component of the database.
5. Indexes: data structure that support efficient access to the data.

Chapter 2

A data model is a notation for describing data or information. It has 3 parts:

- Structure of the data: example in C, arrays or struct, or objects. In data models are somewhat higher level than data structure and are sometime referred to as a conceptual model to emphasize the difference in level.
- Operations on the data: limited to queries and modifications. This limitation is not weakness but a strength.
- Constraints on the data: Database data models usually have a way to describe limitations on what the data can be.

Important: two data models of preminent.

1. Relational model, including object-relational extensions.
2. Semistructured-data model, including XML and related standards. (tree or graphs)

Attributes: Columns of a relations. Attributes appear at the tops of the columns. (or we can say the name of columns, describle the meaning of entries in the column).

Schema: The name of a relation and the set of the attributes for a relation is called the schema for that relation (table name). The attributes in a relation schema are a set, not a list.

Tuples: rows, each tuple has one component for each attribute of the relation. We normally use commas to sprate components, and we use paranthese to surround the tuple.

Domains: type of attributes. ex: string, interger etc depend on each attributes that we want to stores.

Key constraints: A set of attributes form a key for relation if we do not allow two tuples in a relation instances to have the same value in all the attributes of the key.

Example: Movies (title, year, length, genre)
no two movies that had both the same title and the same year.

Relations in SQL

1. Stored relations, called tables. → ordinarily - relation that exists in the database, can be modified by changing its tuples, as well as queried.
2. Views are relations defined by computation. They are not stored, but are constructed.
3. Temporary tables. Constructed by the SQL language processor when it performs its job of executing queries and data modifications.

Data types, similar to C program language.

→ Simple Table Declaration.

CREATE TABLE tablename (attribute domain, ...);

→ Modifying Relation schema.

DROP TABLE table_name;

→ table_name is no longer part of the database schema, and we can no longer access any of its tuples.

ADD follow by an attribute name and it data type.

ALTER TABLE MovieStar ADD phone Char(16);

→ Add phone: attribute (colum) with type is CHAR(16)

→ DROP phone; (if want to drop.

DEFAULT value is the value that appears in a column if no other value is known.

ex. ALTER TABLE Movies ADD phone Char(16) DEFAULT 'unlisted';

Declaring Key:

Two ways to declare an attribute or set of attributes to be a key in the CREATE TABLE.

1. Declare one attribute to be a key when that attribute is listed in the relation schema.
2. Add to the list of items declared in the schema.

If the key consist of more than one attribute, we have to use method 2.

→ There are two declarations that may be used to indicate keyness: PRIMARY KEY or UNIQUE.

→ If primary key is used, then attributes are not allowed to have NULL as a value for their components. But NULL is permitted if the set is declared UNIQUE, however.

Relational Algebra

Set operations on Relations.

- R ∪ S, the union, all in R and S
- R ∩ S, intersection, elements in both R and S
- R − S, difference, elements that are in R but not in S → return element in R but not in S

Projection operator is used to product from a relation R a new relation that has only some R's columns.

symbol $\Pi_{\text{title, year}}(\text{Movies})$

Selection applied to a relation R, product a new relation with a subset of R's tuples. (query with argument where)

ex. $\sigma_{\text{length} \geq 100}(\text{Movies})$

Also we can use AND with selection.

$\sigma_{\text{length} \geq 100 \text{ AND name} = \text{'Fox'}}(\text{Movies})$.

Cartesian Product : of two set R and S is the set of pairs that can be formed by chosing the first element of the pair to be any element of R and the second any element of S,

denote $R \times S$

Natural Joins : Pairing only those tuple that match in some way. Denoted $R \bowtie S$

A	B
1	2
3	4

R

B	C	D
2	5	6
4	7	8
9	10	11

S

→

$R \bowtie S$

A	B	C	D
1	2	5	6
3	4	7	8

Theta - Joins based on condition C is $R \bowtie_C S$.

+ Take the product of R and S
+ Select from the product only those tuple that match the condition C.

A	B	C
1	2	3
6	7	8
9	7	8

U

B	C	D
2	3	4
2	3	5
7	8	10

V

A	U.B	U.C	V.B	V.C	D
1	2	3	2	3	4
1	2	3	2	3	5
1	2	3	7	8	10
6	7	8	7	8	10
9	7	8	7	8	10

$U \bowtie_{A<D} V.$

Naming and Renaming. denote $\rho_{S(X,C,D)}(S)$

& $\rho_{RS(A,B,X,C,D)}(R \times S)$

Relationship Among Operations

$$R \cap S = R - (R - S)$$

$$R \bowtie_c S = \sigma_c (R \times S)$$

$$R \bowtie S = \pi_L (\sigma_C (R \times S))$$

Theta-join can be written

$$\sigma_{A<D \text{ AND } U.B \neq V.B} (U \times V)$$

Superkeys: A set of attributes that contains a key called a superkey, short for 'superset of a key'. And it need to satisfy:

1. It functionally determines all other attributes of the relation.
2. A super key need not satisfy the second function minimality.

→ there are can be more than one key in the relation.

3.2.1 Functional Dependencies.

A functional Dependencies (FD) on a relation R is a statement of the form "If two tuples of R agree on all of the attributes $A_1, A_2, \ldots A_n$ then they must also agree on all another list of attributes $B_1, B_2, \ldots B_m$. We write this FD formally as

$A_1 A_2 \ldots A_n \rightarrow B_1, B_2 \ldots B_n$ and say that

→ _____ functional determine $B_1, B_2, \ldots B_m$.

Chapter 4.

Entity / Relationship model → E/R Model Variation,

1. A primitive type, as in the version presented here,
2. A "struct" as in C, or tuple with a fixed number of primitive components.
3. A set of values of one type: either primitive or a "struct" type.

E/R Diagrams is a graph representing entity sets, attributes, and relationships.

+ Entity sets are represented by rectangles.
+ Attributes are represented by ovals
+ Relationships are represented by diamonds.

Edges connect an entity to its attributes and also connect a relationship to it entity set.

Entity/Relationship Model → Database (contain all tables)
The arrow pointing to entity set indicates that each movie is owned by at most one studio.

Weak Entity Sets

1. Sometimes entity sets fall into a hierachy based on classifications unrelated to the "is a hierachy".
2. These entity sets often have no attributes of their own. Their key is formed from the attributes that are the key attributes for the entity sets they connect

→ It must use foreign keys in conjunction with its attributes to create a primary key. In nother hand, it can't be uniquely identified by it attributes alone.

Chapter 2,3,4 is about E/R, Relational Algebra, and Functional Dependencies.

Part II. Relational Database Programming.

+ **Bags:** Allow the same tuple to appear more than once in a relation.
+ Bags can also use Union, Intersection, and Difference of Bags but.
  + R ∪ S, tuple t appears n + m times.
  + R ∩ S, tuple t appears min(n, m) times.
  + R − S, tup t appear max(0, n − m) times.

+ **Selection on Bags.**

A	B	C
1	2	5
3	4	6
1	2	7
1	2	7

$\sigma_{C \geq 6}(R) \Rightarrow$

A	B	C
3	4	6
1	2	7
1	2	7.

Duplicate Elimination on Bags or Unique in SQL denoted $\delta(R)$

Ex.

A	B
1	2
1	2
3	4

$\delta(R) \rightarrow$

A	B
1	2
3	4.

Aggregation Operations

A functions. SUM(att), AVG(att), MIN, MAX, COUNT. Is also applied to bags relation.

Extending the Projection Operation.

A	B	C
0	1	2
0	1	2

$\Pi_{A,\ B+C \rightarrow X}(R) \rightarrow$

A	X
0	3
0	3
3	9

Chapter 6. Database Language SQL.

Projection in SQL.

```
Select title AS name, length AS duration
From Movies
Where studioName = 'Disney' AND year = 1990;
```

* AS will be like rename the column and display it as new name.

→ We can use SELECT clause.

↳ SELECT title AS name, length * 0.016667 AS lengthInHours

then it will take every single values in length, multiply it with 0.016667 then save it into lengthInHours

Formular.

SELECT L
FROM R ↔ $\pi_L(\sigma_C(R))$
WHERE C

* Pattern Matching in SQL

SQL provide the capability to compare string on the basis of a simple pattern match. Expression is,

s LIKE p. where s is a string and p is a pattern. Only use two special characters [% and _]

Ex: WHERE title LIKE 'Star____';

the last 4 can be any thing Ex Star Wars, Star Trek.

WHERE Title LIKE '%' 's%';

'a%' → any value start with a
'%a' → ——— end with a
'%or%' → any value that have 'or' in any position.
'_r%' → Finds any value that r in the second position.
'a_%_%' → ——— starts with 'a' and at least 3 char in length.
'a%o' → starts with 'a' and ends with 'o'.

* NULL Values and Comparisons Involving NULL
+ Any operator with NULL like x or +, the result will be NULL.
+ When we compare anything with NULL value → result will be UNKNOW.

p. 255. for Truth table for three-valued logic.

* Ordering the Output.
ORDER BY < list of attributes >

ORDER BY clause follows the WHERE clause and any other clauses. The ordering is performed on the result of the FROM, WHERE, and other clause.
Also, optional for ORDER BY is DESC, ASC
→ descending and ascending.
→ Also can be ordering by several column columns.

* We can also use INTERSECT for SQL.
(Select name, address From MovieStar Where gender = 'F')
INTERSECT
(Select name, address From MovieExec Where netWorth > 10000k)
→ Will product table name, address of star who is F and have netWorth > 10000k.

* EXCEPT.
(Select name, from star) EXCEP (Select name from Mova)
→ Give the name of movie stars who are not also movie executives regardless of gender or net worth.

Subqueries. One query can be used in various ways to help in the evaluation of another. A query that is part of another is called a subquery.

1. Subqueme can returns a single constant, and this constant can be compared with another value in a WHERE clause.
2. Subquires can return a relations that can be used in various way in WHERE clauses
3. Subquire can also appear in FROM clause, follows by a tuple variable that represents the tuples in the result of the subquery.

Using IN after WHERE to compare multiple values that return from subqueries

Example with multiple subqueries.

```
SELECT name
FROM MovieExec
WHERE cert# IN
    ( SELECT producerC#
      FROM Movies
      WHERE (title, year) IN
          (SELECT movieTitle, movieYear
           FROM StarsIn
           WHERE starName = 'Harrison Ford'
          )
    );
```

This finding the producers of Harrison Ford's movies. For the compare WHERE clause, we can use

>, <, =

also we can use NOT EXISTS from a subquery.

* SELECT DISTINCT statement is used to return only distinct (different) values.

* Using JOIN in SQL, we have to use ON.

```
SELECT title, year, length, genre, studioName, ...
FROM Movies JOIN StarsIn ON
        title = movieTitle AND year = movieYear;
```

However, for Outerjoins. We use.

MovieStar NATURAL FULL OUTER JOIN MovieExec;

——— LEFT, etc.

* The cost of Duplicate Elimination.
One might be tempted to place Distinct after every Select, on the ~~beer~~ theory that it is harmless. In fact, it is very expensive to eliminate duplicate from a relation. It have to sorted or partitioned so that identical tuples appear next to each other.
Sorting the relation to find duplicate will cost the running time. (Need a best algorithm for sorting this.

Sorting Algorithms.

	Worst-Case	Avg/expect case.
Insertion sort	$\Theta(n^2)$	$\Theta(n^2)$
Merge sort	$\Theta(n\lg n)$	$\Theta(n\lg n)$
Heapsort	$O(n\lg n)$	—
Quicksort	$\Theta(n^2)$	$\Theta(n\lg n)$ expected.
County Sort	$\Theta(k+n)$	$\Theta(k+n)$
Radix Sort	$\Theta(d(n+k))$	$\Theta(d(n+k))$
Bucket Sort	$\Theta(n^2)$	$\Theta(n)$ avg-case.

* Count(*) to get the number of tuples in relation.
* Count(distinct name), we do not count the duplicate.

* The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns.

```
SELECT column_name(s)
FROM table_name
WHERE condition.
```

```
GROUP BY column_name
ORDER BY column_name.
```

DESC → high to low;
ASC → low to high;

HAVING Clause

The having clause was added to SQL because the WHERE keyword could not be used with aggregate functions.

Example: HAVING MIN(year) < 1930

DATABASE MODIFICATIONS

* Insert

INSERT INTO R(A_1, .. A_n) VALUES (v_1, ..., v_n);

INSERT INTO table_name (colum1, .. colum$_n$)
VALUES (value1, ... value$_m$).

Also we can use INSERT INTO SELECT to copies data from one table and inserts it into another table.

INSERT INTO table 2
SELECT * FROM table1
WHERE conditions;

or we can select colum to colum.

* Deletion

DELETE FROM table
WHERE <condition>;

* Update.

UPDATE table
SET colum = value, colum2 = value2 ...
WHERE <conditions>;

ex. Update instructor
set salary = salary + 10000
where salary <= 50 000;

Timing of Insertions.

The SQL standard requires that the query be evaluated completely before any tuples are inserted.

Transactions in SQL

Serializability:

Example: two customer trying to book the same seat simultaneously at the same time.

→ The problem is solved in SQL by the notion of a 'transaction', which is informally a group of operations that need to be performed together.

→ One common approach is for the DBMS to lock elements of the database so that two functions can't access them at the same time.

(one type queries and other SQL state

+ When using the generic SQL interface, each statement is a transaction by itself. However, SQL allows the programmer to group several statements in a single transaction. The SQL command START TRANSACTION is used to mark the beginning of a transaction. There two way to end a transaction.

1. Use COMMIT clause.
2. ROLLBACK clause the transaction to abort, or terminate unsuccessfully.

4 levels.

+ serializable

Isolation Level: (the transaction must appear to run either completely before or completely after each other transaction).

+ repeatable-read: every tuple read in response to a query will appear if the query is repeated

+ read-committed: Only tuples written by transactions that have already committed may be seen by the transaction.

+ read-uncommitted: No constant on what the transaction may see.

~~Chapter 7~~

Chapter 7. Constraints and Triggers

+ Active element is an expression or statement that we write once and store in the database, expecting the element to execute at appropriate times.

+ foreign-key constraint

* Declaring Foreign-Key Constraints

A foreign key constraint is an assertion that values for certain attributes must make sense.

1) The referenced attribute of the second relation must be declared UNIQUE or the PRIMARY KEY for their relation. Otherwise, we can't make the foreign-key declaration

2) Values of the foreign-key appearing in the first relation must also appear in the referenced attributes of some tuple.

a) REFERENCES <table> (<attribute>).

b. FOREIGN KEY (<attributes>) REFERENCES <table>
(<attributes>)

Example:

Studio (name, address, presC#).

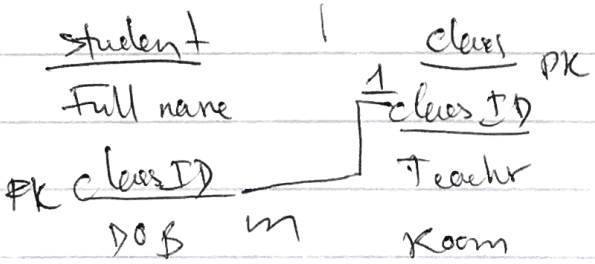
The primary key is name and which has a foreign key presC# that references cert# at relation.

MovieExec (name, address, cert#, netWorth).

1). CREATE TABLE Studio (
name CHAR(30) PRIMARY KEY,
address VARCHAR (255),
presC# INT REFERENCES MovieExec(cert#)

2) CREATE TABLE Studio
name CHAR(30) PRIMARY KEY,
address VARCHAR (255)
presC# INT,
FOREIGN KEY (presC#) REFERENCES
MovieExec (cert#)

In another word, A foreign key is a column or set of columns in one table, that uniquely identifies rows in another table.



1 students have 1 class but 1 class can have many student.

If the primary key of a table is a foreign key of another table, you can't delete ~~that~~ tuples of the table unless you delete the ~~tuples~~ values from the foreign table first.

Exam. (PK)

student(ID, name, dept_name, tot_cred)

takes (ID, course_id, sec_id, semester, year, grade)

(foreign key)

takes.ID is foreign key of student.ID.

Foreign key use as a key to access data from ~~two~~ cross table.

Rules to delete

PK

* So when we insert, we have to insert to student first then insert to takes after (FK)
* ~~Same~~ However, when we delete, we have to delete from the takes (FK) first before we can delet student. PK

Both PK and FK can't be NULL.

* A tuple with a foreign key values that does not appear in the referenced relation is said to be a dangling tuple.
  → it can't be in a join of its relation with the referenced relation.

* We can declear two kinds of constrands within SQL CREATE TABLE statement.
  1. A constaunt on a single attribute.
  2. A constraint on a tuple as a whole.

Constraints is rules for data in table. ex. NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK, DEFAULT, INDEX.

* SQL Check Constraint.

This use to limit the value range that can be placed in a column.

```
CREATE TABLE Person ( ID int NOT NULL,
                      Name varchar (255) NOT NULL,
                      Age INT CHECK (Age >= 18));
```

→ Single. this rule let you can't inter value age < 18.

for multiple columns. we use. can use both.

```
create table persons ( id int not null, name varchar(255) NOT NULL,
    age int, city varchar(255)
    Constraint CHK_Person CHECK (Age >= 18 AND City = 'S'));
```

→ We use ALTER to add constraint to the table already that created.

```
ALTER TABLE Persons ADD CHECK (Age >= 18);
```

To allow CHECK Constraint on multiple columns. we use.

→
```
Alter Table Persons Add Constraint CHK_PersonAge CHECK
    (Age >= 18 AND City = 'S');
```

Remove constraint use DROP CONSTRAINT CHK_PersonAge;

Modification of Constraints

In order to modify or delete an existing constraint, it is necessary that the constraint have a name. To do so, we precede the constraint by the keyword CONSTRAINT and a nomame for the constraint.

* **Assertions** Prefer as a function, use to check argument.

Formular → CREATE ASSERTION <assertion-name> CHECK (<condition>);

- We can declare an assertion as an element of a database schema. The declaration gives a condition to be checked.

Assertions are checked whenever there is a change to one of the relation & involved. Attribute- and tuple-based checks are only checked when the attribute or relation to which they apply changes by insertion or update.

~~ER~~

Trigger SQL

~~Trigger~~

```
CREATE TRIGGER FixYearTrigger
BEFORE INSERT ON Movies
REFERENCING
    NEW ROW AS NewRow
    NEW TABLE AS NewStuff
FOR EACH ROW
WHEN NewRow.year IS NULL
UPDATE NewStuff SET YEAR = 1915;
```

The SQL standard includes triggers that specify certain events that awaken them. Once awakened, a condition can

be checked, and if true, a specified sequence of actions will be executed.

Chapter 8. Views and Indexes.

Virtual View, create a virtual views of a relation (table)

CREATE VIEW <view_name> AS <view_definition>;

EX. Create View test as select * from student;

After we create View, we can use DROP, UPDATE, DELETE, INSERT, same as normal tbl, but it doesn't saved to database table.

If the View is created by Arithmetic Expression (+, −, ×, ÷) then the View can't be update. Also, what with Aggregate function, it can't be update or GROUP BY

Aggregation function: Min, Max, Count, Avg, Mode, Median, Sum ...

View is a data object which does not contain any data. Its contains are the resultant of the base table.

When we update/delete/drop/insert into View, the based table also will be updated.

Indexes in SQL

An index of an attribute A of a relation is a data structure that makes it efficient to find those tuples that have a fixed values for attribute A. We could think of the index as a binary search tree of (key, value) pairs;

BST: Left child smaller than parent;
Right child larger than parent.

Note: the key for the index can be any attribute or set of attributes, and need not be the key for the relation on which the index is built.

Data structure used by a typical DBMS is the 'B-tree' which is a generalization of a balanced binary tree

Why indexes?

When the relations are very large, it becomes expensive to scan all the tuples of a relation to find those tuples that match a given condition.

→ Indexes may also be useful in queries that involve a join.

→ To retrieve data from database very fast. User can't see the index, they just use to speed up search/queries.

Declaring Indexes

```
CREATE INDEX YearIndex ON Movies(year);
```

or

```
CREATE INDEX KeyIndex ON Movies(title, year);
```

```
DROP INDEX YearIndex;
```

Some Useful Indexes.

1) Queries in which a value for the key is specified are common. Thus, an index on the key will get used frequently.

2). Since there is at most one tuple with a given key value, the index returns either nothing or one location for a tuple. Thus, at most one page must be retrieved to get that tuple into main memory.

* SQL Server Database, have 128 pages per megabyte. each pages begin with a header.

* The typical relation is stored over many disk blocks (pages) and the principal cost of a query or modification is often the number of pages that need to be brought to main memory. The index help us for fast query. However, the indexes themselves have to be stored, at least partially, on disk, so accessing and modifying the indexes themselves cost disk accesses. → It requires one disk access to read a page, another disk access to write the changed page, ⇒ is about twice as expensive as accessing the index or the data in a query.

* Materialized Views are disk based and are ~~updated~~ update periodically based ~~on~~ upon the query definition. While View are virtual only and run the query definition each time they are accessed.

Chapter 9. SQL in Server Enviroment.

Database is the main container, it contains the data and log files, and all the schemas within it. You always back up a database, it is a discrete unit on its own.

Schema are like folders within a database, and are mainly used to group logical objects together, which leads to ease of setting permissions by schema.

Simple Code in C:

```
Exec SQL begin declare section;
    char studioNam[50], studioAddr[256];
    char SQLSTATE[6];
Exec SQL end declar section;
EXEC SQL Insert Into Studio (name, address)
    Value (:studioName, :studioAddr);
```

* Stored Procedures PSM.

Persistant, Stored Modules. Is a part of the latest revision to the SQL standerd, called SQL:2003.

PSM function.

```
CREATE PROCEDURE <name> (<parameters>)
    <local declarations>
    <procedure body>;
```

Loop in PSM.

```
REPEAT
    <STATEMENT List>
until <condition>
end repeat.
```

```
LOOP
    <statement list>
END LOOP;
```

```
WHILE <condition> DO
    <statement List>
END WHILE;
```

Chapter 10 Security in SQL

Using Grant to give permission to user.

Nested-relational ~~model~~ model, we allow attributes of relations to have a type that is not atomic; in particular, a type can be a relation schema

Ex. Atomic type: Int, real, string, etc.. can the the type of an attribute.

Ex of nested

Stars (name, address (street, city), birthdate,
movies (title, year, length))

↳ :

Movies (title, year, length)
Stars (name, address (street, city), birthdate,
movies ({* Movies}))

Recursion in SQL.

WITH R AS <definition of R> <query involving R>

Ex. With recursive Reaches (frm, to) AS
(select frm, to from flights)
Union
(select R1. frm, R2. to
from Reaches R1, Reach R2
Where R1. to = R2. frm)
Select * from reaches;

* Describe table_name show how the table construct.

2nd Book.

DATABASE PROCESSING

A trigger is a procedure that is executed when a particular data activity occurs. In general it use to make sure everything is OK before we update or insert to the database.

- SELECT *
  FROM EMPLOYEE
  WHERE Department NOT IN ('Accounts', 'Finance').
  or
  IN
- GROUP BY after WHERE
- INTERSECT. Between 2 select and usually inside a return subquery ('WHERE _ IN(SELECT _ INTERSECT SELECT_
- JOIN, will depend on version can brand of SQL so they will have the difference syntax, but the principle is the same.
- (INNER) JOIN: Return records that have matching value in both table
- LEFT (OUTER) JOIN: Return all records from the left table and the matched record from the right table.
- RIGHT (OUTER) JOIN: Same with left, but take the right table.
- FULL (OUTER) JOIN: Return all records when there is a match in either left or right table.

- INSERT INTO table_name VALUES (' ', ' ',).
- UPDATE table_name SET column = value WHERE column = value
- DELETE FROM table_name WHERE <condition>

- CREATE VIEW view_name AS
  SELECT colums_view AS colums for view
  FROM table_name.

- By using ~~select~~ create view, we can use to combine two or more different colum into 1.

Exam. SELECT Name, ('('+ AreaCode +')' + PhoneNumber) As phone...

SQL in program code.

OpenSQL (Select * FROM CUSTOMER)
Move curser to first row;
While curser not past end of table {
set custName = Curser.Name;
... statement that use custName ...
Advance cursor to next row;
};
... continue processing ...

Use Trigger for Validity Checking. SET TRIGGERSERVER ON
This is an example how to create trigger.

?. Create or update trigger display
Before delete or insert or update on instructor
For each row when (new.ID > 0)
Declare sal_diffe number;
Begin
SalaDiffe := :NEW.Salary - :OLD.Salary;
dbms_output.put(' ' || :old.salary);
end;

Why need Transcation:

To make sure when we insert or deletes all the information from a tuples have to ~~be~~ ~~cop~~ completely made.

Example. buying item in a stock and only have 100 items left. Need to make sure not override or not engough in stock.

We use the loge lock / release / wait state. to solve this.

Book Principles of Database Systems.

Chapter 5. Improving the Quality of Database Design.

FD. Functional Dependencies between Attributes.

A → B

FD

Example. Customer (accountID, lastname, firstname, street, city, state, zipcode)

S: zipcode → {city, state}

{street, city, state} → zipcode

Zipcode determines city and state. Same with street, city, state.

Inferring Functional Dependencies

Rule 1: Reflexivity: If X ⊇ Y, then X → Y

Rule 2: Augmentation: If X → Y, then XZ → YZ.

Rule 3: Transitivity: If X → Y and Y → Z, then X → Z

Rule 4: Decomposition: If X → YZ, then X → Y

5: Union: If X → Y and X → Z, then X → YZ

Rule 6: Pseudo-transitivity: If $X \to Y$, and $WY \to Z$, then $WX \to Z$

2NF (Second normal form). A schema that has no non-prime attribute that is partially dependent on Key.

3NF (third normal form). A schema that is in 2NF and has no non-prime attribute that is transitively dependent on a Key of the schema.

+ Left side of any FD must be super key or last level candidate key

Loss less-join decomposition if

$R_1 \cap R_2 \to R_1$ or $R_1 \cap R_2 \to R_2$

A decomposed $\{R_1, R_2\}$

1) $X \supseteq Y$ then $X \to Y$
2) $X \to Y$ then $XZ \to YZ$
3) $X \to Y$ and $X \to Z$ then $X \to Z$
4) $X \to YZ$ then $X \to Y$
5) $X \to Y$ and $X \to Z$ then $X \to YZ$
6) $X \to Y$ and $WY \to Z$ then $WX \to Z$.

1NF.

An attribute of a table cannot hold multiple values. It should hold only atomic values (single value).

2NF.

+ Table is in 1NF
+ No non-prime attribute is dependent on the proper subset of any candidate key of table.

3NF.

+ Table is in 2NF
+ All fields can be determined only by the key in the table and no other column.
(No transitive functional dependency)

+ $\alpha \rightarrow \beta$ is a trivial function dependency
+ α is a superkey for R
+ Each attribute A in $\beta - \alpha$ is contained in a candidate key for R.

* Trivial functional dependency (that is, $\beta \subseteq \alpha$)
→ $\alpha \rightarrow \beta$

Armstrong's Axioms (certain these 3 rules)

+ Reflexivity: If $X \supseteq Y$, then $X \rightarrow Y$
+ Augmentation: If $X \rightarrow Y$ then $XZ \rightarrow YZ$
+ Transitivity: If $X \rightarrow Y$, and $Y \rightarrow Z$ then $X \rightarrow Z$

SQL to summary sales number and price by store and date.
along with the hierarchies on store and date.

Sales, store, date_info

```
select store_id, city, state, country, date, month, quarter, year
       sum(number), sum(price)
from sales, store, date_inter
where sales.date = date_inter.date and sales.store_id = store.store_id
group by rollup (country, state, city, store_id),
         rollup (year, quarter, month, date)
```

1) Find document at least k of a given set of n keywords.

2) PageRank is used to measure the popularity of a page based on the popularity of pages that link to the page.

Based on the Page

+ 5 people in the team.
+ Working with other in UK, Holland, do alot of work.
+ Working for high level stuff.
+ Product → non-manager product. Move to the cloud.
+ → Google searchy → create contain, system, search, find
→ Have junior guys for 2-3 years.

```
array1 = [a, b, c]
array2 = array1;
array.length = 0;   array1.length = 0.
Console
```

```
var a = function()
{
console
.fun b = { log.b }.
```

```
var a = function( conso.lg.a)
function b (.consolo.lg b)
```

A → C
A → E
BC → D
BC → E
D → A

A → C ⇒ $(A)^+ = \{AE\}$ ✓
A → E ⇒ $(A)^+ = \{AC\}$ ✓
BC → D ⇒ $(BC)^+ = \{BCE\}$ ✓
BC → E → $(BC)^+ = \{BCDAE\}$ redant
D → A → $(D)^+ = \{D\}$

min → A → C , A → E, BC → D , D → A

→ Prime dependencies
(A, C) (A, E) (B, C, D) (D, A)

→ Since BC is in

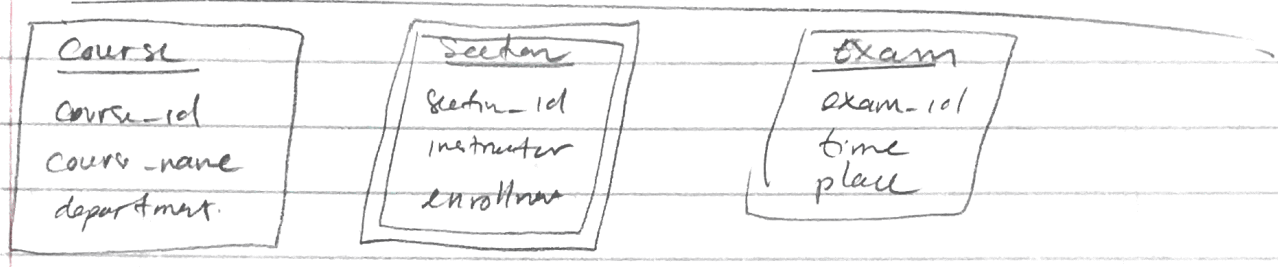
Dell

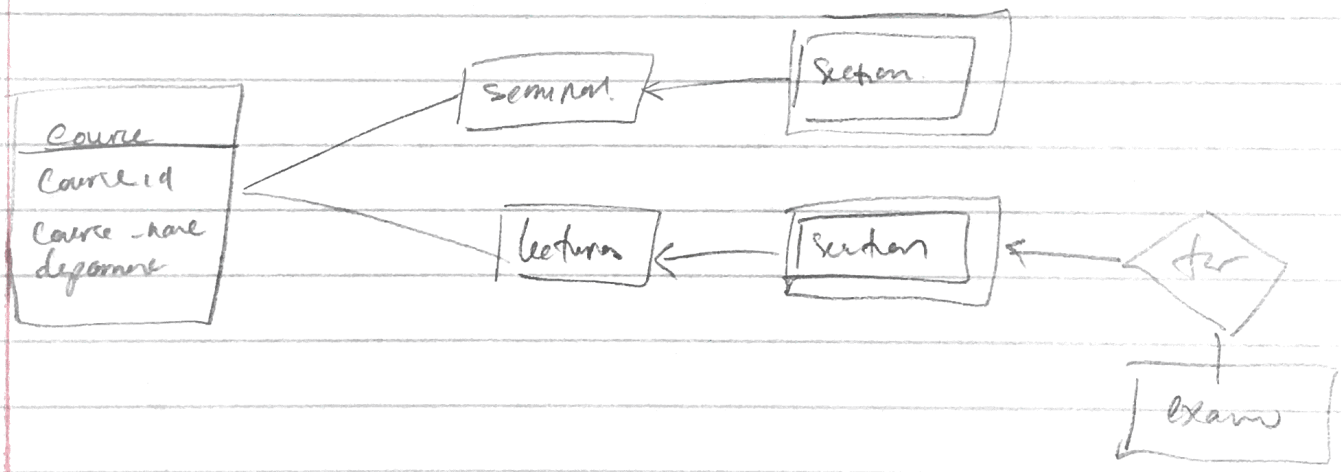
Revature

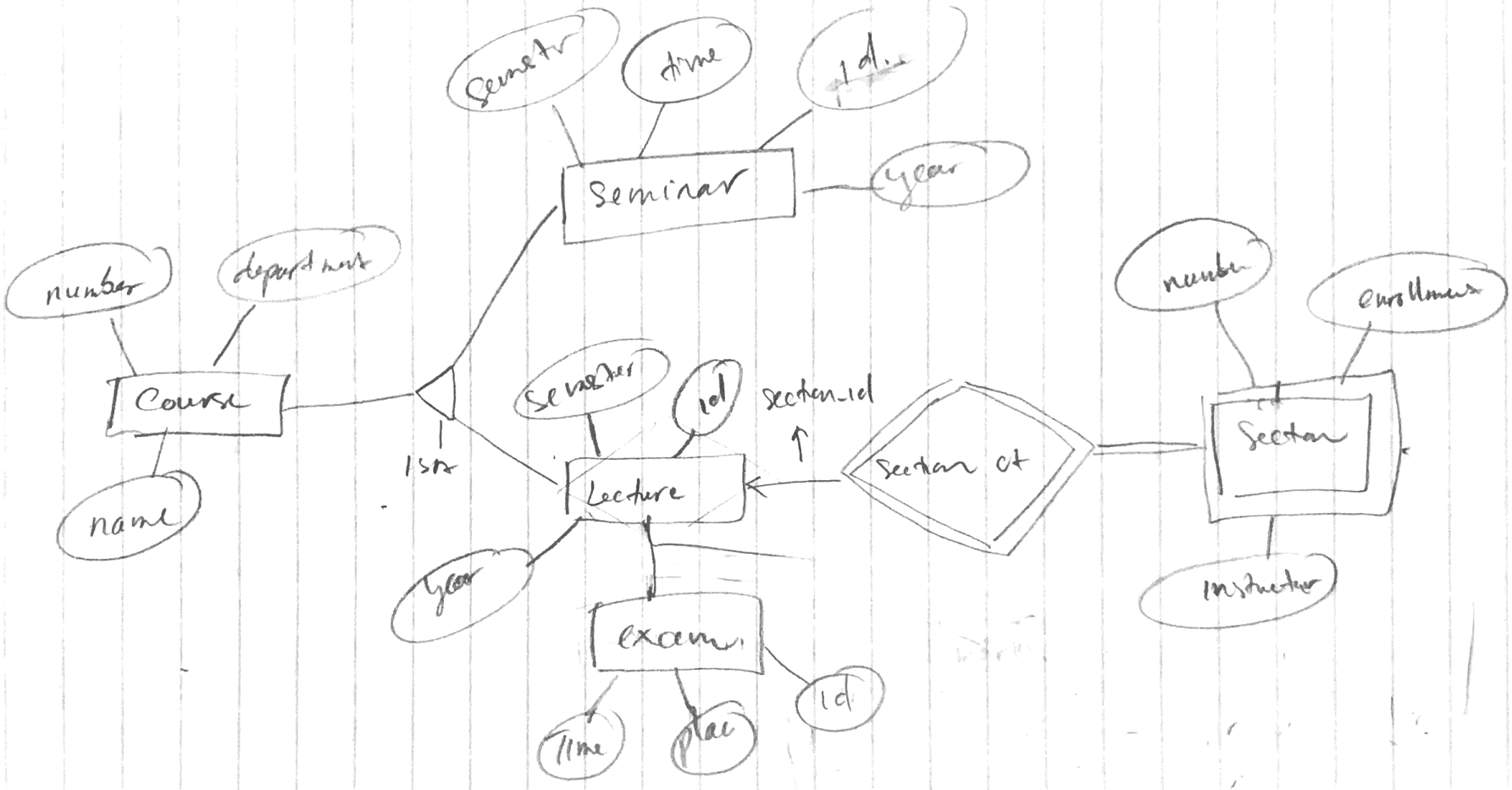
RKT

Verint

Revature







Course (id, name, department)

Lecture (exam_id, section, semester, year)

seminar (semstr, id, time, place

section (sect_id, enrollment, Ath

Lecture (lectr_id, year, semester)

exam (lecture_id, time, place,

3 NF. Defines

for all FD on F^+ of form $\alpha \rightarrow \beta$ where $\alpha \subseteq R$ and $\beta \subseteq R$. At least one of follows hold.

$\alpha \rightarrow \beta$ is a trivial FD in R.

α is a super key of R

For each attribute A in $\alpha \rightarrow \beta$ is contained in candidate