VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui

EXAMINATIONS — 2004
MID-YEAR

COMP 302

Database Systems

Time allowed: 3 Hours

Instructions: Answer all questions.
Make sure that your answers are clear and to the point.
Calculators and printed foreign language dictionaries are allowed.
No reference material is allowed.
There are 180 marks on the exam.

CONTENTS:

Appendices:

A. Formulae for Computing Query Cost Estimate
B. SQL reference

Question 1. Relational Data Model and Relational Algebra [26 marks]

(a) What is a Relation in the Relational Model, and what two properties must its elements have? **[3 marks]**

ANSWER

(b) What is the difference between a Relation and a Relation Schema? **[3 marks]**

ANSWER

(c) What is a "key" in the Relational Model? **[3 marks]**

ANSWER

(d) What is a "foreign key" in the Relational Model? **[3 marks]**

ANSWER

(e) Suppose R = ({A,B,C},{A}) and S = ({C,D,E},{C}) are relation schemas, and r(R) and s(S) are relations over R and S respectively.

(i) Write a relational algebra expression for the relation consisting of those tuples in r(R) for which C > 10, restricted to the attributes A,B. **[2 marks]**

ANSWER

(ii) Write a relational algebra expression for the relation over {B,C,D} consisting of the values of B, C, and D from all pairs of tuples from r(R) and s(S) that have the same value for the attribute C. **[2 marks]**

ANSWER

(iii) What kind of join did you use in **(ii)**? **[1 marks]**

ANSWER

(f) Suppose r_1 and r_2 are relations over relation schemas $N_1(\{A, B\}, \{A\})$ and $N_2(\{B, C\}, \{B\})$, respectively. Using the relational algebra project operator, express the referential integrity constraint $N_1[B] \subseteq N_2[B]$. **[2 marks]**

ANSWER

(g) Consider the following set of three relation schemas and referential integrity constraints:

$N_1(\{A, B\}, \{A\})$	$N_2[A] \subseteq N_1[A]$,
$N_2(\{A, C, D\}, \{AC\})$	$N_3[A] \subseteq N_1[A]$
$N_3(\{A, C, E, F\}, \{ACE\})$	$N_3[(A, C)] \subseteq N_2[(A, C)]$

(i) Which of the three referential integrity constraints given above is a consequence of the other two constraints? **[2 marks]**

ANSWER

(ii) Use the transitivity property of the set inclusion operator to show that the referential integrity constraint you identified in part (i) will be always satisfied if the other two referential integrity constraints are satisfied. **[5 marks]**

ANSWER

Question 2. SQL **[26 marks]**

The three relation schemas below are part of a relational database schema to record details of tests measuring the time it takes for a test car to stop on different road surfaces when fitted with different tyres. The tyres are specified by their make and model. The database records the width and tread pattern of each tyre, the gravel-size and kind of tar of each road surface, and the stopping time of each test.

Tyre ({Make, Model, Width, TreadPattern }, {Make+Model })

RoadSurface ({SurfaceCode, GravelSize, Tar } {SurfaceCode })

StoppingTest ({SurfaceCode, Make, Model, Time }, {SurfaceCode+Make+Model })

The attribute constraints are given in the following table:

Attribute	**Data Type**	**Max. Length (bytes)**	**Null**
Make	Char	variable length	No
Model	Char	8	No
Width	Decimal	6	No
TreadPattern	Char	15	Yes
SurfaceCode	Char	6	No
GravelSize	Int	4	No
Tar	Char	Variable length	Yes
Time	Integer	4	No

(a) Write an SQL statement to define the RoadSurface table. **[4 Marks]**

ANSWER

(b) Write an SQL statement to return the make and model of all tyres with a width greater than 15. **[3 Marks]**

ANSWER

(c) Write an SQL statement to return a list of all tyres and their stopping time on roads with a gravel size of 5, ordered from the best tyres (fastest to stop) to the worst. **[4 Marks]**

ANSWER

(d) Write an SQL statement that would define the StoppingTest table. Include any key and referential integrity constraints. **[7 Marks]**

Note, deleting a tyre or a road surface from the database should also remove any stopping tests involving the tyre or the road surface. Modifying data in the tyre or road surface relations should be possible.

ANSWER

(e) Write an SQL statement to return the make and model of each tyre that has ***not*** been tested on all possible road surfaces, along with the number of road surfaces that the tyre has not been tested on. **[8 marks]**

ANSWER

Question 3. Enhanced Entity Relationship Data Model [25 marks]

Construct an EER diagram for each of the situations described below. The names of the entities are in ***italic*** font. Your diagrams should indicate the keys of all entities, as well as the cardinality and participation constraints of all relationships.

(a) An HR database. [4 marks]

The HR section of a company keeps a database of all ***Applicants*** for positions. It identifies each Applicant by an application number, and stores their name and address (house number, street, and town)

ANSWER

(b) A Factory Database [5 marks]

A factory has a large number of milling machines and a staff of machine operators. Each milling ***Machine*** is identified by a machine number, and also has a machine type. Each ***Operator*** is identified by their name (they use nicknames if two people have the same real name) and has a seniority level.

There are several working sessions in a day. At the beginning of the day, the manager assigns each operator to a machine for each session. An operator may have a different machine in each session. There are more machines than there are operators.

ANSWER

(c) A Storage Database **[8 marks]**

A Geology department keeps its collection of rocks in large cabinets.

- Each ***Cabinet*** is identified by the name of the researcher who collected the rocks in the cabinet. (No one has more than one cabinet).
- Each cabinet contains a set of ***Drawer****s* where the rocks are put. The drawers of a cabinet are numbered 1, 2, 3, …
- Each drawer is labeled with the field trip or trips where the rocks were collected (each field trip is specified by a place and date).

ANSWER

(d) A Mouse Database. **[8 marks]**

A Psychology department has a database of the mice that they have bred. They use the male mice for maze experiments.

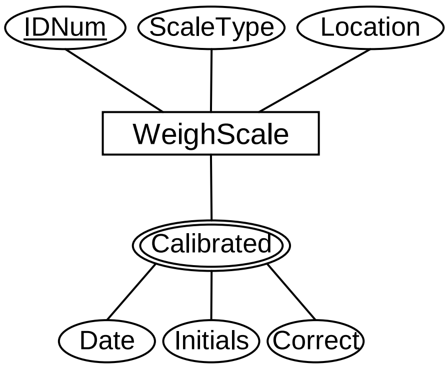
- Each ***Mouse*** is identified by a name, and is described by its sex and date of birth.
- Each ***FemaleMouse*** has a count of her children, and her largest litter size.
- Each ***MaleMouse*** has a count of the number of experiments it has done, its total score, and its average score.
- The database must also record which two mice are the parents of each mouse.

ANSWER

Question 4. Mapping EER to relational data model [25 marks]

For each of the following EER diagrams, map the diagram to an equivalent relational database schema. Specify the attributes and keys of the relation schemas, any attribute constraints, and a set of non-redundant referential integrity constraints. You do not need to specify the domains of the attributes.

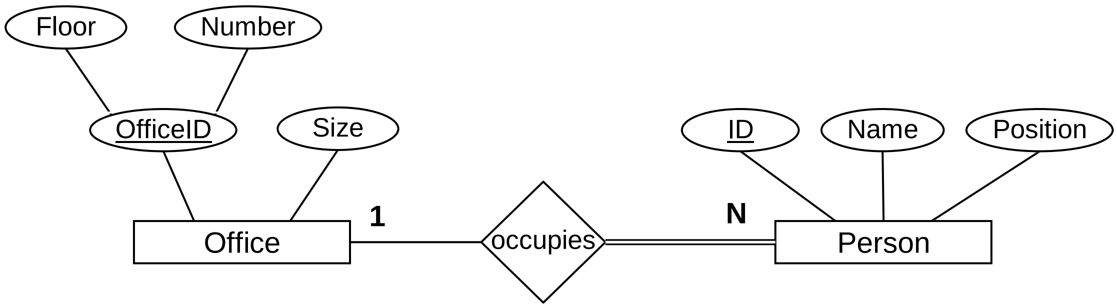
(a) A database of all the measuring scales in a research facility and a record of their calibration history. **[5 Marks]**



Note: No WeighScale is calibrated more than once a day.

ANSWER

(b) A database of the allocation of staff to offices.



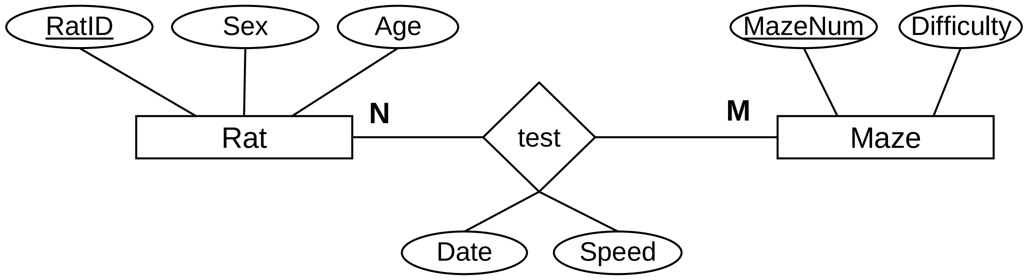
(i) Give your relational database schema: **[4 marks]**

ANSWER

(ii) Explain how the relationship with total participation is enforced in your relational database schema. **[2 marks]**

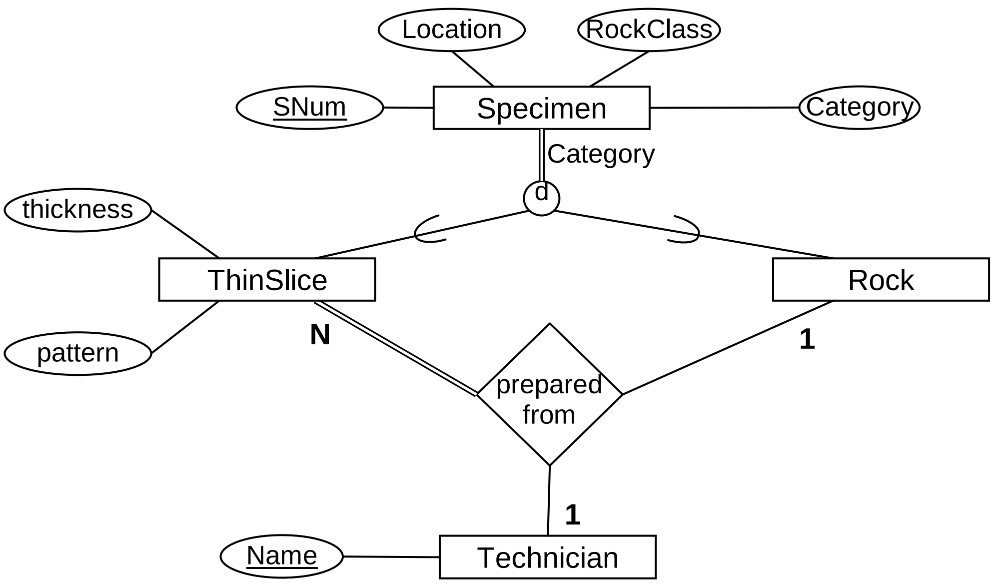
ANSWER

(c) A database of experiment results. **[5 Marks]**
A rat may not be tested on the same maze more than once.



ANSWER

(d) The specimens in the Geology Department collection are either rocks or thin slices prepared from the rock specimens. Your schema design should not require any null attributes.



(i) Give your relational database schema: **[4 marks]**

ANSWER

(ii) In the EER diagram, the specialisation is total. Explain why your relational database schema does or does not enforce this constraint. **[2 marks]**

ANSWER

(iii) Express the additional constraints that are present in the EER but not in your relational database schema and explain what would be required to enforce them: **[3 marks]**

ANSWER

Question 5. Functional Dependencies and Normalization [33 marks]

(a) Consider the following set of functional dependencies: **[6 marks]**

$$F_1 = \{AB \to C,\quad DEG \to H,\quad A \to C,\quad DE \to G\}.$$

Transform the set F_1 into a new set of fd's F_1', in which each fd is left reduced (*i.e.* has no redundant attributes on its left hand side).

ANSWER

(b) Consider the following set of left reduced functional dependencies: **[4 marks]**

$$F_2 = \{AB \to C,\quad C \to D,\quad AB \to D\}.$$

Transform the set F_2 into a new, non redundant set of fd's F_2'.

ANSWER

(c) Suppose (U, F) is a universal relation schema, where **[7 marks]**

$$U = \{A, B, C, D\} \text{ and } F = \{A \to B,\quad C \to B,\quad D \to B\}.$$

Suppose that starting from (U, F) the following set of BCNF relation schemas has been produced:

$$S = \{N_1(\{A, B\}, \{A\}),\quad N_2(\{C, B\}, \{C\}),\quad N_3(\{D, B\}, \{D\})\}.$$

If you consider that S is a lossless join decomposition of (U, F), explain why it is.
If you consider that S is not a lossless join decomposition of (U, F), explain why it is not, and transform it into a new set of relation schemas S' that will be at least in BCNF and a lossless join decomposition of (U, F).

ANSWER

(d) Consider the following relation schema

$$N(\{A, B, C, D\},\quad \{AB \rightarrow C,\quad C \rightarrow A,\quad D \rightarrow B,\quad AB \rightarrow D\})$$

(i) Find all relation schema N keys. **[4 marks]**

ANSWER

(ii) What is the highest normal form that relation schema N is in? Justify your answer. **[4 marks]**

ANSWER

(e) The Inference Rules for functional dependencies are given in Figure 5.1. You have seen them in the lectures and in the textbook.

Given U, F, and $X, Y, Z, V, W \subseteq U$
1. (Reflexivity) $Y \subseteq X \;\vdash\; X \rightarrow Y$ (trivial fd)
2. (Augmentation) $X \rightarrow Y$ and $W \subseteq Z \;\vdash\; XZ \rightarrow YW$
3. (Transitivity) $X \rightarrow Y$ and $Y \rightarrow Z \;\vdash\; X \rightarrow Z$
4. (Decomposition) $X \rightarrow YZ \;\vdash\; X \rightarrow Y$ and $X \rightarrow Z$
5. (Union) $X \rightarrow Y$ and $X \rightarrow Z \;\vdash\; X \rightarrow YZ$
6. (Pseudo transitivity) $X \rightarrow Y$ and $WY \rightarrow Z \;\vdash\; WX \rightarrow Z$

Figure 5.1

State whether each of the following implications is true or false. If it is false, give a relation with only two tuples that satisfies the functional dependencies in the left-hand side of the implication but does not satisfy the dependencies in the right-hand side. If it is true, show which inference rules you used to prove it is true.

(i) $V \rightarrow X$ and $VX \rightarrow Y \;\vdash\; V \rightarrow Y$ **[4 marks]**

ANSWER

(ii) $X \rightarrow Z$ and $Y \rightarrow Z \;\vdash\; X \rightarrow Y$ **[4 marks]**

ANSWER

(Spare Page for extra working)

Question 6. Query Optimisation **[25 marks]**

Table 1 below contains a description of a part of the *LTSA* database schema (as in question 2). Table 2 below contains the corresponding parameters of the physical database structure.

Attribute	Data Type	Max. Length	Null	Default
Relation name: *Tyre*, Primary Key: *Make* + *Model*				
Make	char	15	N	
Model	char	8	N	
Width	decimal	6,2	N	
TreadPattern	Char	15	N	
Relation name: *RoadSurface*, Primary Key: *SurfaceCode*				
SurfaceCode	char	6	N	
GravelSize	int	4	N	
Tar	char	15	N	
Relation name: *StoppingTest*, Primary Key: *SurfaceCode* + *Make* + *Model*				
SurfaceCode	char	6	N	
Make	char	15	N	
Model	char	8	N	
Time	int	4		

Table 1

Relation schema	*L* tuple length	*F* blocking factor	*r* number of tuples	*x* primary key index height	Number of B* tree leaf nodes
Tyre	*44*	*11*	*3,000*	*3*	*120*
RoadSurface	*25*	*20*	*500*	*2*	*10*
StoppingTest	*33*	*15*	*150,000*	*5*	*10,000*

Table 2

(a) Draw the heuristic optimization tree that corresponds to the SQL query: **[5 marks]**

```
SELECT *
FROM (RoadSurface r NATURAL JOIN StoppingTest s NATURAL JOIN Tyre t)
WHERE t.Make = "Goodyear";
```

ANSWER

(b) Draw the heuristic optimization tree that corresponds to the SQL query: **[5 marks]**

```
SELECT Make, AVG(Time)
FROM StoppingTest
GROUP BY Make;
```

ANSWER

For parts (c) and (d), assume:

- The *StoppingTest* relation contains data about tests of all *3000* tyres on only *50* surfaces,
- The intermediate results of the query evaluation are materialized,
- The final result of the query is materialized,
- The size of each intermediate or final result block should not exceed *500* bytes,
- There is a buffer pool of *6000* bytes provided for query processing in the main memory,
- There are primary indexes on *Tyre*.(*Make* + *Model*), *RoadSurface*.(*SurfaceCode*), and *StoppingTest*.(*SurfaceCode* + *Make* + *Model*) available,
- Primary indexes are B* trees with nodes of size at most *500* bytes.

NOTE:
Some of the formulae you may need when computing the estimated query costs are given at the end of this exam paper.

(c) Calculate the lowest execution cost of the query **[7 marks]**

SELECT * FROM RoadSurface NATURAL JOIN StoppingTest;

ANSWER

(d) The following SQL statement retrieves SurfaceCodes that have been used so far to test tyres **[8 marks]**

```
SELECT DISTINCT SurfaceCode
FROM StoppingTest
```

Find the lowest cost estimate of the query above.

Hint: If there is an index on the queried relation that is sorted by the attribute list of a SELECT DISTINCT clause, it can be used to evaluate SELECT DISTINCT operation without accessing the queried relation itself. Also note, in a B* tree index, all primary key values of a relation are stored in the leaf nodes, leaf nodes are linked into a linked list, and there is a pointer to the leftmost leaf node.

ANSWER

(Spare page for extra working)

Question 7. Concurrency Control **[20 marks]**

(a) In the lectures, we discussed the following four kinds of transaction anomalies: **[6 marks]**

- Dirty Read,
- Lost Update,
- Phantom Record, and
- Unrepeatable Read.

For each of the following descriptions, state which kind of anomaly it describes:

(i) A transaction T_1 reads a database item A. Another transaction T_2 reads the same database item A. The transaction T_1 updates A and commits. Then T_2 updates the database item A and commits.

ANSWER

(ii) A transaction T_1 reads a database item A that is updated by a not committed transaction T_2 and then T_2 aborts.

ANSWER

(iii) A transaction T_1 reads a database item A. Another transaction T_2 updates A and commits. Then T_1 reads the database item A again.

ANSWER

(iv) A transaction T_1 locks database items that satisfy a selection condition and updates them. During that update, another transaction T_2 inserts a data item that satisfies the same selection condition. After that, transaction T_1 reads and displays all database items that satisfy the selection condition and a user discovers a not updated item.

ANSWER

(b) Two transaction programs "Borrow_Book" and "Return_Book" are given in Figures 7.1 and 7.2, respectively. They use a database whose schema is given in Figure 7.3. These programs were written in SQL using some of the PostgreSQL specific commands we discussed in tutorials and you used in your Project 2. Suppose the two programs are run concurrently as shown in Figure 7.4. After the "Return_Book" program issued the third SQL statement, both programs were stalled. **[14 marks]**

Borrow_Book

```
BEGIN;

SELECT * FROM Customer
  WHERE CustomerId = 007 FOR UPDATE;

SELECT Copies_Left FROM Book
  WHERE isbn = 1111 FOR UPDATE;

INSERT INTO Cust_Book VALUES(007, 111, '2004-05-31', '2004-06-19');

UPDATE Customer SET BooksBorrowed = BooksBorrowed + 1;

UPDATE Book SET CopiesLeft = CopiesLeft - 1;

COMMIT;
```

Figure 7.1

Return_Book

```
BEGIN;

SELECT * FROM Book
  WHERE isbn = 1111 FOR UPDATE;

SELECT * FROM Customer
  WHERE CustomerId = 007 FOR UPDATE;

DELETE FROM Cust_Book WHERE CustomerId = 007 AND isbn = 1111;

UPDATE Customer SET BooksBorrowed = BooksBorrowed - 1;

UPDATE Book SET CopiesLeft = CopiesLeft + 1;

COMMIT;
```

Figure 7.2

Customer({*CustomerId*, *Name*, *BooksBorrowed*}, {*CustomerId*})
Book({*isbn*, *Title*, *NoCopies*, *CopiesLeft*}, {*isbn*})
Cust_Book({*CustomerId*, *isbn*, *BorrowDate*, *DueDate*}, {*CustomerId* + *isbn*})

Figure 7.3

(i) Why were the programs stalled? **[2 marks]**

ANSWER

Time	Borrow_Book	Return_Book
	BEGIN;	
	SELECT * FROM Customer WHERE CustomerId = 007 FOR UPDATE;	
		BEGIN;
		SELECT Copies_Left FROM Book WHERE isbn = 1111 FOR UPDATE;
	SELECT Copies_Left FROM Book WHERE isbn = 1111 FOR UPDATE; //Here, the program was stalled	
		SELECT * FROM Customer WHERE CustomerId = 007 FOR UPDATE; //Here, the program was stalled

Figure 7.4

(ii) What will PostgreSQL do to resolve the problem? **[2 marks]**

ANSWER

(iii) Rewrite the first three statements of either of the programs to avoid the anomalous situation you identified in question **(i)**. Your rewriting of the program should not allow any transaction anomaly to happen. **[10 marks]**

ANSWER

* * * * * * * * * * * * * * * * * * *

Formulae for Computing Query Cost Estimate

Blocking factor: $f = \lfloor B / L \rfloor$

Number of blocks: $b = \lceil r / f \rceil$

Selection cardinality of the attribute A : $s(A) = r / d(A)$, where $d(A)$ is the number of different A values

Number of buffers $n = \lfloor K / B \rfloor$, where K is the size of the buffer pool

C (project) = $b_1 + b_2$

C(project_distinct) = $b_1 + b_2 + 2b_2(1 + \lceil log_m\ b_2 \rceil) + b_2 + b_3$, $m = n - 1$

C (select_linear) = $b_1 + \lceil s / f \rceil$

C (select_sec_key) = $x + s + \lceil s / f \rceil$

Costs of join algorithms

(o stands for the outer loop relation, and i stands for the inner loop relation)

C (nested_join) = $b_o + b_i \lceil b_o / (n - 2) \rceil + \lceil js * r_o * r_i / f \rceil$

C (single_join) = $b_o + r_o * f(index_i) + \lceil js * r_o * r_i / f \rceil$ (minimum of 4 buffers required)

C (sort_join) = $b_1(3 + 2\lceil log_m\ b_1 \rceil) + b_2(3 + 2\lceil log_m\ b_2 \rceil) + \lceil js * r_1 * r_2 / f \rceil$, $m = n - 1$

C(partition_join) = $3(b_1 + b_2) + \lceil js * r_1 * r_2 / f \rceil$, $n \geq \lceil (3 + (1 + 4b_1)^{1/2})/2 \rceil$, $b_1 < b_2$

C (sort) = $2b(1 + \lceil log_m\ b \rceil)$

f ($index$):

primary index: $f(index) = x + 1$

secondary index: $f(index)$: $x + s$

Approximate formulae for choosing the most efficient join algorithm

Mandatory condition: $b_o << b_i$

(o stands for the outer loop relation, and i stands for the inner loop relation)

C(single) < C(nested) $\Leftrightarrow r_o * f(index_i) < b_i \lceil b_o / (n - 2) \rceil$ // Providing $r_o * f(index_i) >> b_o$

C(single) < C(sort) $\Leftrightarrow r_o * f(index_i) < b_i(3 + 2\lceil log_m b_i \rceil)$ // Providing $r_o * f(index_i) >> b_o$

C(single) < C(partition-hash) $\Leftrightarrow r_o * f(index_i) < 3b_i$ // Providing $r_o * f(index_i) >> b_o$

C(nested) < C(sort-merge) $\Leftrightarrow b_o < (n - 2)(3 + 2\lceil log_m b_i \rceil)$

C(nested) < C(partition-hash) $\Leftrightarrow \lceil b_o / (n - 2) \rceil \leq 3$

C(sort-merge) < C(partition-hash) $\Leftrightarrow \perp$

Simplified PostgreSQL documentation:

CREATE TABLE

CREATE TABLE *table_name* (
  { *column_name data_type* [DEFAULT *default_expr*] [*column_constraint* [, ...]]
  | *table_constraint* } [, ...]
)

where *column_constraint* is:

[CONSTRAINT *constraint_name*]
{ NOT NULL | NULL | UNIQUE | PRIMARY KEY | CHECK (*expression*) |
  REFERENCES *reftable* [(*refcolumn*)] [ON DELETE *action*] [ON UPDATE *action*] }

table_constraint is:

[CONSTRAINT *constraint_name*]
{ UNIQUE (*column_name* [, ...]) |
  PRIMARY KEY (*column_name* [, ...]) |
  CHECK (*expression*) |
  FOREIGN KEY (*column_name* [, ...]) REFERENCES *reftable* [(*refcolumn* [, ...])]
    [ON DELETE *action*] [ON UPDATE *action*] }

and *action* is one of RESTRICT, CASCADE, SET NULL, or SET DEFAULT

SELECT

SELECT [ALL | DISTINCT]
  * | *expression* [AS *output_name*] [, ...]
  [FROM *from_item* [, ...]]
  [WHERE *condition*]
  [GROUP BY *expression* [, ...]]
  [HAVING *condition* [, ...]]
  [{ UNION | INTERSECT | EXCEPT } [ALL] *select*]
  [ORDER BY *expression* [ASC | DESC | USING *operator*] [, ...]]
  [FOR UPDATE [OF *tablename* [, ...]]]

where *from_item* can be:

[ONLY] *table_name* [*] [[AS] *alias* [(*column_alias_list*)]] |
(*select*) [AS] *alias* [(*column_alias_list*)] |
from_item [NATURAL] [*join_type*] JOIN *from_item* [ON *join_condition* | USING (*join_column_list*)]

and *join_type* can be:

INNER |
LEFT [OUTER] |
RIGHT [OUTER] |
FULL [OUTER] |
CROSS
For INNER (the default) and OUTER join types, exactly one of NATURAL, ON *join_condition*, or USING (*join_column_list*) must appear. For CROSS JOIN, none of these items may appear.

CREATE VIEW

CREATE VIEW *view* [(*column name list*)] AS SELECT *query*

Some Data Types

integer, int, smallint
character[*n*], char[*n*], character varying[*n*], varchar[*n*], varchar
numeric, numeric[*precision*], numeric[*precision, scale*], real, double
boolean, date,

Note: [*xxx*] means *xxx* is optional, { *xxx* | *yyy* } means *xxx* or *yyy*.