SQL – Quick Reference Guide

# Introduction

## What is an RDBMS?

Relational Database Management System.

Examples: Oracle, SQL Server, SYBASE, Informix, DB2, MS Access.

Relational database systems store data in the form of inter-related tables.

## Who works with Database Systems?

### Operations - Keep the DBMS Running

#### Database Administrator

A database administrator (short form DBA) is a person responsible for the design, implementation, maintenance and repair of an organization's [database](http://en.wikipedia.org/wiki/Database).

##### Duties

* Transferring Data
* Replicating Data
* Maintaining database and ensuring its availability to users
* Maintaining the data dictionary
* Controlling privileges and permissions to database users
* Monitoring database performance
* Database backup and recovery
* Database security

### Database Development - New Projects

#### Database Analysts

**Overview:** Assists in planning, designing, and implementing the Common Database. Such activities involve interaction with development and end-user personnel to determine application data access requirements, transaction rates, volume analysis, and other pertinent data required to develop and maintain integrated databases. Works under general supervision, assists in analysis and design activities associated with the development and maintenance of the database. Must be competent to work independently in most phases of database management. Reports to the Manager, Data Management. Additional information available includes essential job functions, additional responsibilities, and education and experience requirements.

#### Data Architect

#### Database Modeler

#### Programmer Analyst

#### Tester

### Users

#### Data Analysts

The data analyst's job description frequently includes importing, cleaning, transforming, validating or modeling data with the purpose of understanding or making conclusions from the data for decision making purposes.

The data analyst's job description may include presenting data in charts, graphs, tables, designing and developing relational databases for collecting data and in some organizations it also includes building or designing data input or data collection screens.

### Managers

Chief Information Officer (CIO)

Systems Manager

## ANSI Standard SQL

SQL is (mostly) the same on all database systems.

# Data Integrity

There are three aspects of data integrity: Entity Integrity, Referential Integrity, and Domain Integrity.

## Entity Integrity

This is actually the easiest type of integrity. We need to ensure that entities (that is, rows) do not get mixed up with each other. Essentially we need to ensure that each row in a table has some marker that uniquely identifies it. In other words, a primary key ensures entity integrity.

## Domain Integrity

We want to ensure that columns accept only the correct type of information. That is, each column has a domain and we want data only from that domain.

### Data type

The data type alone goes a long way to ensure domain integrity.

### NULL/NOT NULL

This is actually a constraint on a column although we don’t declare this constraint the same way.

### CHECK CONSTRAINT

A CHECK CONSTRAINT provides a simple facility to limit the values that can be put into a column.

THIS IS TRUE IN SQL SERVER AND ORACLE.

-- This table will restrict the values in GPA.

--

CREATE TABLE Person

(PersonID int NOT NULL PRIMARY KEY,

PersonName varchar(40) NOT NULL,

GPA number(3,2) NOT NULL CHECK (GPA > 2.5 AND GPA <= 4.0))

drop table Person

go

-- !!!!! The following will not work....\

CREATE TABLE Person

(PersonID int NOT NULL PRIMARY KEY,

PersonName varchar(40) NOT NULL,

EducationTypeID int NULL,

GPA numeric(4,3) NOT NULL CHECK (GPA > 2.5 AND GPA <= 4.0 AND EducationTypeID = 1))

-- a constraint made at the "column level" cannot reference other columns

drop table Person

go

-- the following DOES WORK!!

CREATE TABLE Person

(PersonID int NOT NULL PRIMARY KEY,

PersonName varchar(40) NOT NULL,

EducationTypeID int NULL,

GPA numeric(4,3) NOT NULL,

CONSTRAINT CK\_PersonEdGPA CHECK (GPA > 2.5 AND GPA <= 4.0 AND EducationTypeID = 1))

-- a check constraint at the table level can reference all columns.

### DEFAULT CONSTRAINT

A default sets what the column will contain if the user does not provide a specific value.

Both of these work in Oracle.

CREATE TABLE Person

(PersonID int NOT NULL PRIMARY KEY,

PersonName varchar(40) NOT NULL,

GPA numeric(4,2) DEFAULT 2.0);

CREATE TABLE Persons

(

P\_Id int NOT NULL,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Address varchar(255),

City varchar(255) DEFAULT 'Sandnes'

);

ALTER TABLE Persons

MODIFY City DEFAULT 'New York';

UNIQUE CONSTRAINT

CREATE TABLE Persons  
(  
P\_Id int NOT NULL PRIMARY KEY,  
LastName varchar(255) NOT NULL,  
FirstName varchar(255),  
Address varchar(255),  
City varchar(255),  
CONSTRAINT uc\_PersonName UNIQUE (FirstName,LastName)  
)

Warning: I’m told that ORACLE will not index NULL values. Therefore a column with a UNIQUE constraint that allows null will allow more than one NULL entry.

SYBASE and SQL Server DO index NULL values. Therefore a column with a UNIQUE constraint that allows null will allow only ONE null entry. (Primary keys on the other hand do not allow nulls at all.)

## Referential Integrity

### Actions On the Parent Table

INSERT – no need to check referential integrity

DELETE – need to check whether there are child records.

UPDATE – (specifically update to the primary key) need to check whether there are child records.

### Actions On the Child Table

INSERT – must ensure that the parent record exists

UPDATE – (specifically update to the foreign key) need to check whether the new parent record exists.

DELETE – no need to check whether a parent exists.

### Syntax - ANSI Standard

ALTER TABLE Emp\_tab

ADD (

FOREIGN KEY (Deptno) REFERENCES Dept\_tab (Deptno)

[ON DELETE { **NO ACTION** | CASCADE | SET NULL | DEFAULT }]

[ON UPDATE { **NO ACTION** | CASCADE | SET NULL | DEFAULT }]

;

ON DELETE SET NULL works in Oracle

It appears Oracle has not implemented the ANSI feature of SET DEFAULT.

ALTER TABLE Person

ADD

CONSTRAINT person\_dept\_fk

FOREIGN KEY (DeptID) REFERENCES Dept(DeptID)

ON DELETE SET DEFAULT);

SQL Error: ORA-03001: unimplemented feature

03001. 00000 - "unimplemented feature"

\*Cause: This feature is not implemented.

\*Action: None.

# Data Modeling

## Entities

**Entities** are the "things" about which you are storing information. An entity is stored in the physical database as a **row in a table**.

Tables are made up of rows and columns.

In database theory, a table can also be called a "relation". A table or relation is a set up tuples.

## Attributes

**Attributes** are the specific pieces of information that you are storing for the entities. Attributes are modeled in the physical database as **columns** in the table.

## Rows, Records, and "Tuples".

A row is the complete information about the entity. The terms row and record are often used interchangeably.

In database theory, a row can also be referred to as a "tuple".

## Primary Keys

A primary key uniquely identifies each entity or row in the table.

Database systems enforce that primary key columns cannot be null, and must be unique.

## Foreign Keys

A foreign key is placed on the **child table** which references the primary key on the **parent table**. This is how relationships are modeled.

## One-to-Many Relationships

One to many relationships involve a **parent table** and a **child table**.

### Maximum Cardinality

Maximum Cardinality is usually shown on entity relationship diagrams. It represents the maximum number of entities that is possible for the given relationship.

At the "one" end of a one-to-many relationship (the parent table) the maximum cardinality is 1.

At the "many" end of a one-to-many relationship (the child table) the maximum cardinality is "many".

### Minimum Cardinality

Minimum Cardinality is not always shown on entity relationship diagrams. Minimum cardinality represents the minimum number that is allowed for the given relationship.

Minimum cardinality could be one, which would mean that there must be an entry.

Minimum cardinality could be zero, which would mean that there is no entry.

### Identifying and Non-Identifying Relationships

Not all entity relationship diagrams will represent identifying and non-identifying relationships.

In an "identifying" relationship, the child entity "takes its identity" from the parent entity. That is, the primary key of the parent table is part of the primary key of the child table. On some entity relationship diagrams, an identifying relationship is represented by a solid line.

In "non-identifying" relationships, the child entity has a separate identity of its own. That is, the child table has its own primary key and the foreign key is not part of the primary key. On some entity relationship diagrams, a non-identifying relationship is represented by a dotted line.

## Many-to-Many Relationships

Many-to-many relationships need to be resolved.

The relationship is modeled using a linking table or an **associative entity**.

## Intuitive Approach to Data Modeling

Step 1: Make a list of nouns (whether they are entities or attributes will sort itself out in time).

Step 2: Clearly define your terms. (It is important to clear up confusion or disagreement about what exactly things mean.)

Step 3: Make an ER diagram adding entities one by one. If the relationship between entities isn’t clear then try to label the connection with a verb.

# Data Definition Language - DDL

CREATE, ALTER, DROP, TRUNCATE

## Data Integrity (Constraints)

### Entity Integrity

For entity integrity, we do not want to get one of our "things" mixed up with another one of our "things". Entity integrity is enforced by PRIMARY KEYS

### Relational Integrity

Relational integrity is enforced by FOREIGN KEYS.

### Domain Integrity

For domain integrity, we want to enforce that values that are entered into columns are from the valid domain.

Domain Integrity is enforced by:

DATATYPES

NULL/NOT NULL

CHECK CONSTRAINT

UNIQUE CONSTRAINT

DEFAULT CONSTRAINT

# Data Control Language - DCL

GRANT, REVOKE, DENY

# Data Manipulation Language - DML

## SELECT

### Columns

For production quality code, FDM would like you to explicitly list the columns, rather than use the asterisk \*.

#### Column aliases

The keyword AS is optional but FDM would like you to use it.

### FROM

#### [ INNER ] JOIN

If you just say “JOIN”, it defaults to “INNER JOIN”. In an inner join, the only rows that are returned are those that have data in both tables. There must be a match.

#### { LEFT | RIGHT | FULL } [ OUTER ] JOIN

If you say “LEFT” or “RIGHT” or “FULL” then the join is an OUTER join.

In a “LEFT” join all rows from the “left” table are returned whether there is a match in the “right” table. If a match can be found then it is joined with the row from the left table.

In a “RIGHT” join all rows from the “right” table are returned whether there is a match in the “left” table. If a match can be found then it is joined with the row from the right table.

A “FULL” outer join contains the rows from the “INNER JOIN” (matches) and includes the extra rows from the “left” table where no matches can be found in the right table and includes the extra rows from the “right” table where no matches can be found in the left table.

#### CROSS JOIN

A CROSS JOIN, also known as a CARTESIAN JOIN or a CARTESION PRODUCT shows every possible combination of rows from the left and right tables. Therefore, if the left table has 4 rows and the right table has 5 rows then the output of the cross join will have 20 rows. (4 x 5 = 20).

#### Table aliases

### WHERE

AND

OR

NOT

BETWEEN x AND y

IN (hardcoded list)

(col1, col2) IN ((a,b),(c,d),(e,f)) -- doubles

LIKE %, \_

### GROUP BY

GROUP BY is used together with data aggregation functions: COUNT, MAX, MIN, AVG, SUM, STDDEV, and VARIANCE

When an aggregation formula is used, and column that is NOT inside an aggregation function must be mentioned in the GROUP BY.

### HAVING

An aggregation function cannot appear in a WHERE clause. The following example will not work: WHERE count(\*) > 1

An aggregation function can appear only in a HAVING clause. The following will work: HAVING count(\*) > 1

When a query is processed:

1. the WHERE is used to filter the raw (ungrouped) records
2. the GROUP BY is applied to summarize and aggregate the raw rows into groups
3. the HAVING clause is applied to keep only the groups which match the criteria
4. lastly any ORDER BY is applied to sort the final output

### ORDER BY

ORDER BY is used to sort the final output.

When it is used, the ORDER BY is always the last clause in the query.

### SET Operations

#### UNION

UNION can be used to take rows from two queries and put them together in a single list of DISTINCT rows. Duplicate rows are removed.

The output from the two queries must have the same number of columns and the same datatypes, respectively.

#### UNION ALL

UNION ALL can be used to take rows from two queries and put them together in a single list of ALL rows.

#### INTERSECT

INTERSECT can be used to take rows from two queries and returns only those rows which are common to both

#### MINUS

MINUS returns all rows from the first query that are not repeated in the second query.

## INSERT

INSERT INTO table\_name [ ( column1,...) ] VALUES (va1,...);

or

INSERT INTO table\_name SELECT ...;

## UPDATE

UPDATE table\_name

SET col1 = ... ,

Col 2 =

WHERE ...

## DELETE

DELETE FROM table\_name

WHERE ...

## Built-in Functions

### Date Functions

Dates are stored internally with some numeric representation (probably floating point) but you will never see it. Whenever you select dates from a database you should explicitly convert using TO\_CHAR. And whenever you put dates INTO a database you should explicitly convert using TO\_DATE.

TO\_CHAR

TO\_DATE

LAST\_DAY

ADD\_MONTHS

MONTHS\_BETWEEN

SYSDATE

### String Functions

SUBSTR()

INSTR()

LTRIM(), RTRIM(), TRIM()

LPAD(), RPAD()

UPPER, LOWER, INITCAP

CONCAT

LENGTH

REPLACE

TRANSLATE

### Numeric Functions

GREATEST, LEAST

ceil, floor, round, trunc

abs, sign

greatest, least

to\_char -used to convert dates or numbers to string

to\_number - used to convert string to numeric

### Decode

Decode is unique to ORACLE.

DECODE(col, 1, 'Hi', 2,'There')

### Case

There are two forms of a CASE statement

#### Form 1:

CASE col

WHEN 1 THEN 'Hi'

WHEN 2 THEN 'There'

END

#### Form 2:

CASE

WHEN col=1 THEN 'Hi'

WHEN col=2 THEN 'There'

END

# Explain Plan

## Indexes

B-tree indexes

Bitmap indexes (outside of the scope of the course)

ANALYZE TABLE

### Indexed search vs. table scan

Covering indexes

## Join Algorithms

### Nested loop

### Hash join

### Merge join

# OLTP vs. OLAP

OLTP means "On Line Transaction Processing". OLTP databases support the day to day recording of data and tend to be highly normalized databases.

OLAP means "On Line Analytical Processing". OLAP databases support data analysis and tend to be multi-dimensional CUBES or star-schema.

Reference: OLTP-OLAP.doc

# Sub queries

## In the WHERE or HAVING Clause

In ORACLE terminology, if a sub-select appears in the WHERE or HAVING clause it is a "subquery.

Subqueries can be used together with: IN, =, EXISTS.

(Outside the scope of this course: sub queries can also be used with > ALL, > ANY, < ALL, < ANY)

## In Line Views

In ORACLE terminology, if a sub-select appears in the FROM or JOIN it is an "in-line view".

## Subqueries as a Column

## Correlated vs. Non-correlated Subqueries

In a correlated subquery, information from the outer query is passed into the subquery. If you try to highlight the subquery to run it alone, it will not work.

In a non-correlated subquery, the subquery can be processed first. If you try to highlight the subquery to run it alone, you can.

## Common Table Expressions (WITH Clause)

A WITH clause is even more like an in-line view that the so-called "in-line views".

# Triggers

Triggers and CHECK constraints play off well against each other. CHECK constraints can provide simple rules of what is and is not allowed in tables. Triggers provide a mechanism to handle more elaborate rules to be enforced by the database.

See the example trigger code.

# PL/SQL

# Understanding Normalization

On an intuitive level, the purpose of normalizing your tables is so that information about entities is stored in one place and one place only. If the same information is stored in multiple places then the risk of making data entry errors is increased and all those places must be kept up to date.

Newcomers to database design sometimes suppose that normalization proceeds in an iterative fashion: a 1NF design is first normalized to 2NF, then to 3NF, and so on. This is not an accurate description of how normalization typically works. A sensibly designed table is likely to be in 3NF on the first attempt.

## First Normal Form (1NF)

### A Simple and Useful Definition

A table is in 1NF if

1. There are no duplicate rows. (The table has a primary key. Primary keys must be unique and do not allow NULL.)
2. All the attributes contain ONE value from the applicable domain. Non-key attributes can be specified to allow NULL or not allow NULL.
3. There are no repeating elements or groups of elements.

### Controversy in the Definition of INF.

#### Atomicity

E.F. Codd's original definition of first normal form made reference to the concept of atomicity. Codd defines an atomic value as one that "cannot be decomposed into smaller pieces by the DBMS.

Controversy:

* A character string would seem not to be atomic, as the RDBMS typically provides operators to decompose it into substrings.
* A fixed-point number would seem not to be atomic, as the RDBMS typically provides operators to decompose it into integer and fractional components.

Chris Date suggests that "the notion of atomicity has no absolute meaning". A value may be considered atomic for some purposes, but may be considered an assemblage of more basic elements for other purposes.

#### "Repeating Groups"

Different theorists have different ideas about this topic. However, the way I see it is repeating groups implies a one-to-many relationship and should be modeled as such, at least for the logical model.

### According to Chris Date

According to Date's definition of 1NF, a table is in 1NF [if and only if](http://en.wikipedia.org/wiki/If_and_only_if) it is "[isomorphic](http://en.wikipedia.org/wiki/Isomorphism) to some relation", which means, specifically, that it satisfies the following five conditions:

1. There's no top-to-bottom ordering to the rows.
2. There's no left-to-right ordering to the columns.
3. There are no duplicate rows.
4. Every row-and-column intersection contains exactly one value from the applicable domain (and nothing else).
5. All columns are regular [i.e. rows have no hidden components such as row IDs, object IDs, or hidden timestamps].

More controversy: To allow null or not allow null. A null able attribute would be in violation of condition 4, which requires every field to contain exactly one value from its column's domain. However, Codd’s later vision of the relational model explicit provision for nulls.

## Functional Dependency

In a table, a column, B, is said to be “functionally dependent” on another column, A, if for every distinct value in the A column there is only one value for B. (In mathematical terms, B is a function of A.)

Notation: A -> B means “B is functionally dependent on A”

More generally, a set of columns, Y, are said to be functionally dependent on a set of columns X if and only if each tuple from X is associated with precisely one tuple in Y. Notation: X -> Y

In the following table, column Y is NOT functionally dependant on column X

|  |  |
| --- | --- |
| X | Y |
| 1 | Kevin |
| 2 | Scot |
| 2 | Jim |

Purpose of all this: if a particular piece of information is functionally dependent on another piece of information then we want to store the fact of that relationship in one place and one place only. We want to avoid redundant information, and reduce the risk of having mistakes in our data.

## Second Normal Form (2NF)

Simple and Useful Definition: Every attribute must be functionally dependent on the primary key. Moreover, in a table that has a composite key, every attribute should be functionally dependent on the WHOLE of the primary key (and not just a part of the primary key).

Formal Statement: “No non-prime attribute in the table is [functionally dependent](http://en.wikipedia.org/wiki/Functional_dependency) on a [proper subset](http://en.wikipedia.org/wiki/Proper_subset) of a [candidate key](http://en.wikipedia.org/wiki/Candidate_key).”

### Problem with Not Adhering to 2NF

If an attribute is dependent on only part of the primary key then it will appear in multiple rows in the table (redundant).

## Third Normal Form (3NF)

Simple and Useful Definition: Every attribute should be functionally dependent on ONLY the primary key (and not on any other column outside of the primary key).

Formal Statement: “Every non-prime attribute is non-transitively dependent on every [candidate key](http://en.wikipedia.org/wiki/Candidate_key) in the table.”

Strictly speaking, a table may have several candidate keys; only one of which will be chosen as the primary key. But if you have a column that is functionally dependent on a column that is not a candidate key for the entire table then that column violates third normal form.

### Problem with Not Adhering to 3NF

Just like 2NF, if your design fails 3NF the same data will be repeated in multiple rows. The goal of normalization is to store data in one place and ONLY one place.

## “Nothing but the Key”

Here is a nice way to remember

“Every non-key attribute must provide a fact about the key, the whole key, and nothing but the key, so help me [Codd](http://en.wikipedia.org/wiki/Edgar_F._Codd)".

This statement sums up 3NF.

## Fourth Normal Form (4NF)

Fourth normal form involves multi-valued dependencies. The problem can arise if you try to model a three-way many-to-many-to-many relationship with a single association table.

4NF was introduced by [Ronald Fagin](http://en.wikipedia.org/wiki/Ronald_Fagin) in 1977.