NANYANG THE INNOVATIVE POLYTECHNIC School of Information Technology

ORACLE SQL Syntaxes

S/N	Name	Syntax
1.	SELECT	SELECT *column1, column2, ...* FROM *table_name*;
2.	WHERE	SELECT *column1, column2, ...* FROM *table_name* WHERE *condition*;
3.	ORDER BY	SELECT *column1, column2, ...* FROM *table_name* ORDER BY *column1, column2, ...* ASC|DESC;
4.	SELECT DISTINCT	SELECT DISTINCT *column1, column2, ...* FROM *table_name*;
5.	AND	SELECT *column1, column2, ...* FROM *table_name* WHERE *condition1* AND *condition2* AND *condition3 ...*;
6.	OR	SELECT *column1, column2, ...* FROM *table_name* WHERE *condition1* OR *condition2* OR *condition3 ...*;
7.	LIKE	SELECT *column1, column2, ...* FROM *table_name* WHERE *columnN* LIKE *pattern*;
8.	IN	SELECT *column_name(s)* FROM *table_name* WHERE *column_name* IN (*value1, value2, ...*);
9.	BETWEEN	SELECT *column_name(s)* FROM *table_name* WHERE *column_name* BETWEEN *value1* AND *value2*;
10.	COLUMN ALIAS	SELECT *column_name* AS *alias_name* FROM *table_name*; *Or* SELECT *column_name "alias_name"* FROM *table_name*;
11.	Case Manipulation Functions	UPPER(*column_name | expresssion*) LOWER(*column_name | expresssion*) INITCAP(*column_name | expresssion*)

<table>
<tr><td>12.</td><td>Character Manipulation Functions</td><td>CONCAT(column1 | expression1, column2 | expression2)

SUBSTR(column_name | expression, starting position, length)

LENGTH(column_name | expression)

LPAD(column_name | expression, total numer of characters in the padded string, the character to pad with)
INSTR (column_name | expression, column_name | expression)</td></tr>
<tr><td>13.</td><td>Additional Manipulation Functions - TRIM</td><td>TRIM([{ { LEADING | TRAILING | BOTH }
[trim_character]
| trim_character
}
FROM
]
trim_source
)

E.g.
SELECT TRIM(LEADING 'a' FROM 'abcba')
FROM DUAL;
Result: bcba</td></tr>
<tr><td>14.</td><td>Additional Manipulation Functions - REPLACE</td><td>REPLACE (string1, string_to_replace, [replacement_string])</td></tr>
<tr><td>15.</td><td>Number Functions</td><td>ROUND (column|expression, decimal places)

TRUNC(column|expression, decimal places)

MOD(column|expression, number)</td></tr>
<tr><td>16.</td><td>DATE FUNCTIONS</td><td><table><tr><td>MONTHS_BETWEEN</td><td>Number of months between two dates
MONTHS_BETWEEN(date_value1, date_value2)</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td></td><td>
<table>
<tr><td>ADD_MONTHS</td><td>Add calendar months to date
ADD_MONTHS(date_value, number)</td></tr>
<tr><td>NEXT_DAY</td><td>Date of the next occurrence of day of the week specified
NEXT_DAY(date_value, weekday)</td></tr>
<tr><td>LAST_DAY</td><td>Last day of the month
LAST_DAY(date_value)</td></tr>
<tr><td>ROUND</td><td>Round date
ROUND(date_value, unit of date)</td></tr>
<tr><td>TRUNC</td><td>Truncate date
TRUNC(date_value, unit of date)</td></tr>
</table>
</td></tr>
<tr><td>17.</td><td>Type Conversion Functions</td><td>TO_CHAR (date column name, 'format model you specify')

TO_CHAR (number, 'format model')

TO_NUMBER (character string, 'format model')

TO_DATE(character string, 'format model')</td></tr>
<tr><td>18.</td><td>NULL Functions</td><td>NVL (expression 1 value that may contain a null, expression 2 value to substitute for null)

NVL2 (expression 1 value that may contain a null, expression 2 value to return if expression 1 is not null, expression 3 value to replace if expression 1 is null)

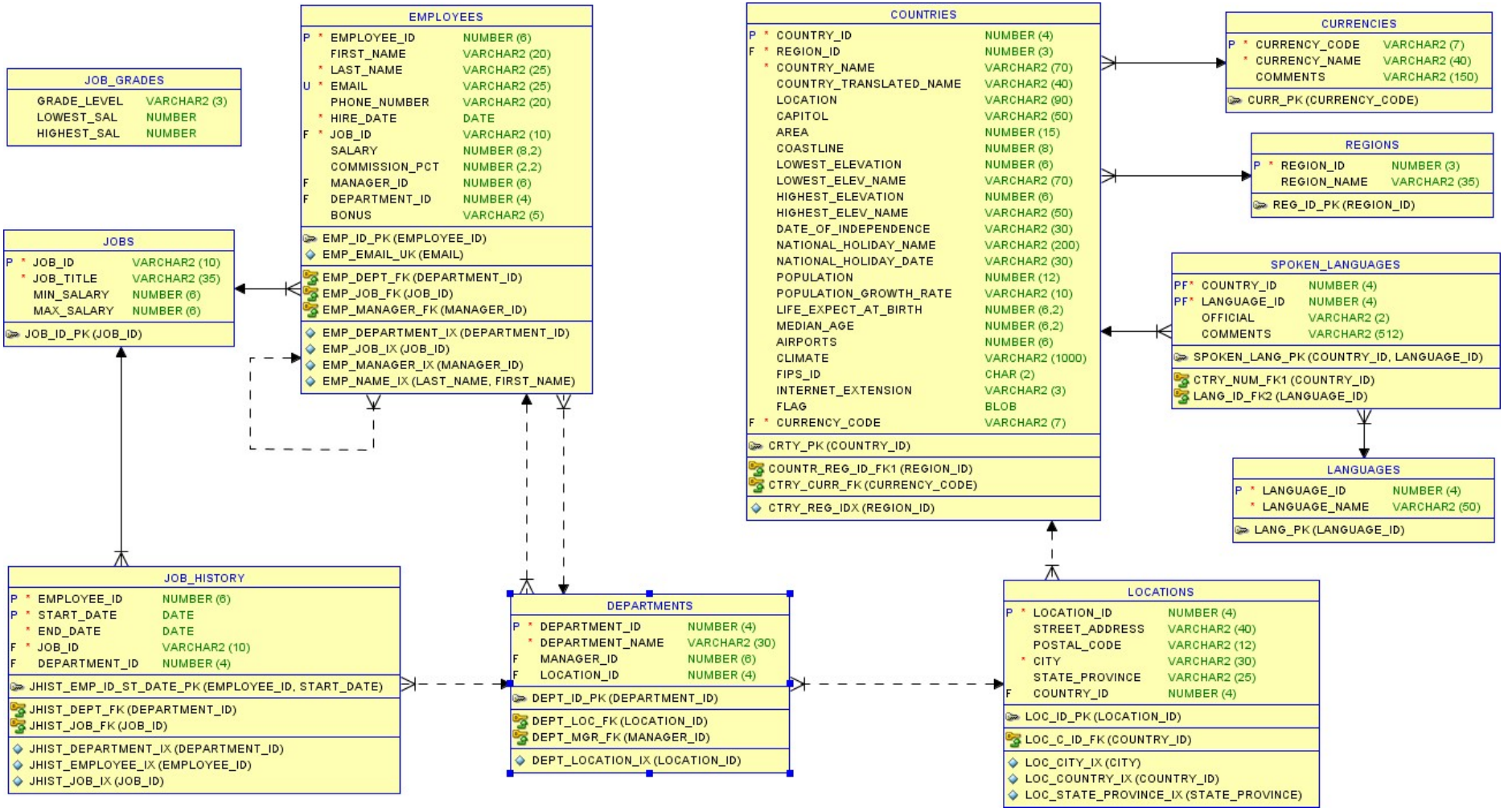
NULLIF (expression 1, expression 2)

COALESCE (expression 1, expression 2, ...expression n)</td></tr>
<tr><td>19.</td><td>Conditional Expressions - CASE</td><td>CASE
WHEN condition1 THEN result1
WHEN condition2 THEN result2
WHEN conditionN THEN resultN
ELSE result
END;</td></tr>
</table>

20.	Conditional Expressions - DECODE	DECODE(column1|expression, search1, result1 [, search2, result2,...,] [, default])
21.	INNER JOIN	SELECT *column_name(s)* FROM *table1* INNER JOIN *table2* ON *table1.column_name = table2.column_name*;
22.	LEFT OUTER JOIN	SELECT *column_name(s)* FROM *table1* LEFT OUTER JOIN *table2* ON *table1.column_name = table2.column_name*;
23.	RIGHT OUTER JOIN	SELECT *column_name(s)* FROM *table1* RIGHT OUTER JOIN *table2* ON *table1.column_name = table2.column_name*;
24.	GROUP BY	SELECT *column_name(s)* FROM *table_name* WHERE *condition* GROUP BY *column_name(s)*;
25.	HAVING	SELECT *column_name(s)* FROM *table_name* WHERE *condition* GROUP BY *column_name(s)* HAVING *condition* ORDER BY *column_name(s)*;
26.		SELECT *column_name(s)* FROM *table_name* WHERE *column_name operator* (SELECT *column_name(s)* FROM *table_name* [WHERE] *condition*)

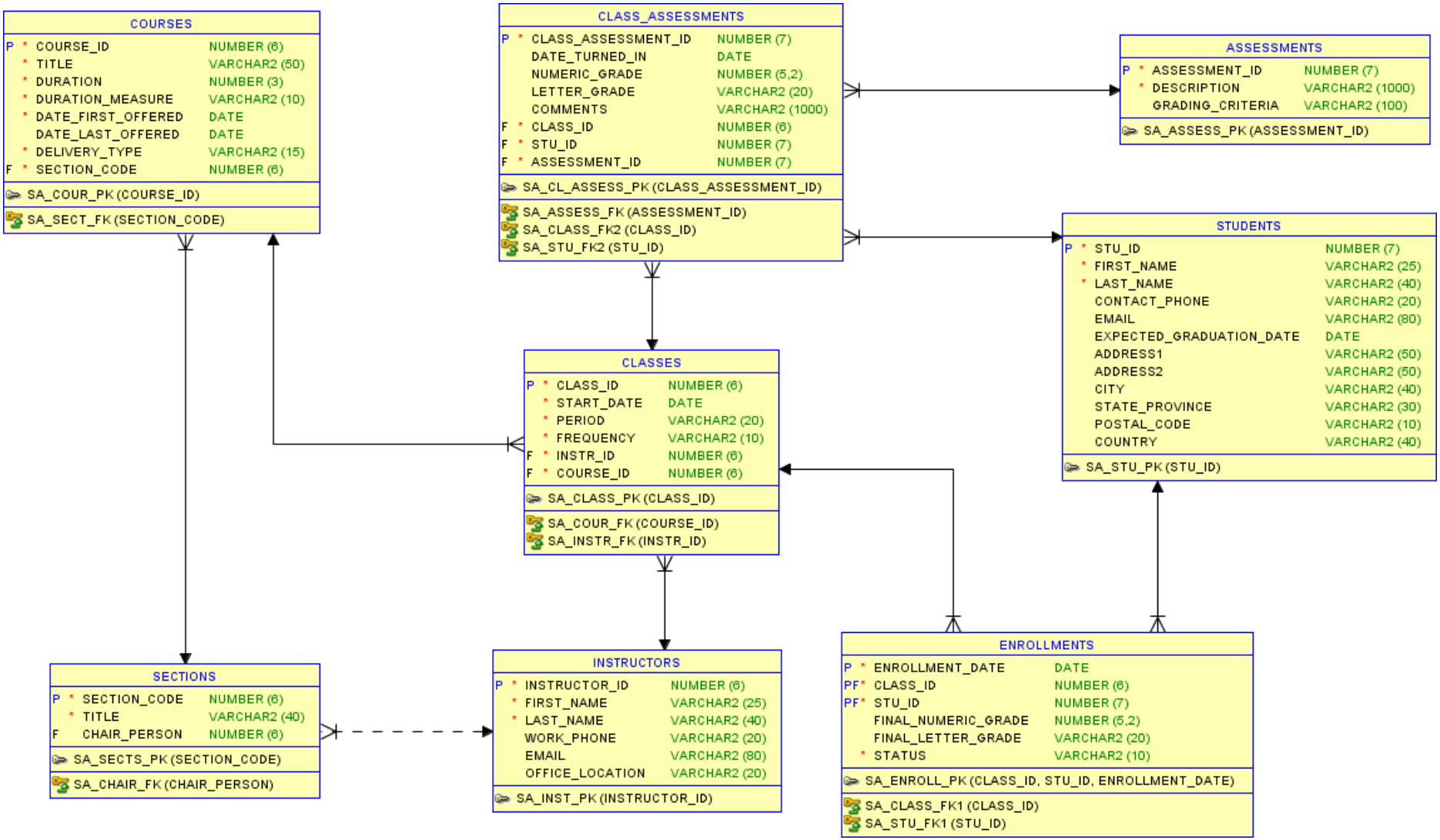
NANYANG THE INNOVATIVE POLYTECHNIC School of Information Technology

Database Schema for Employee/Department



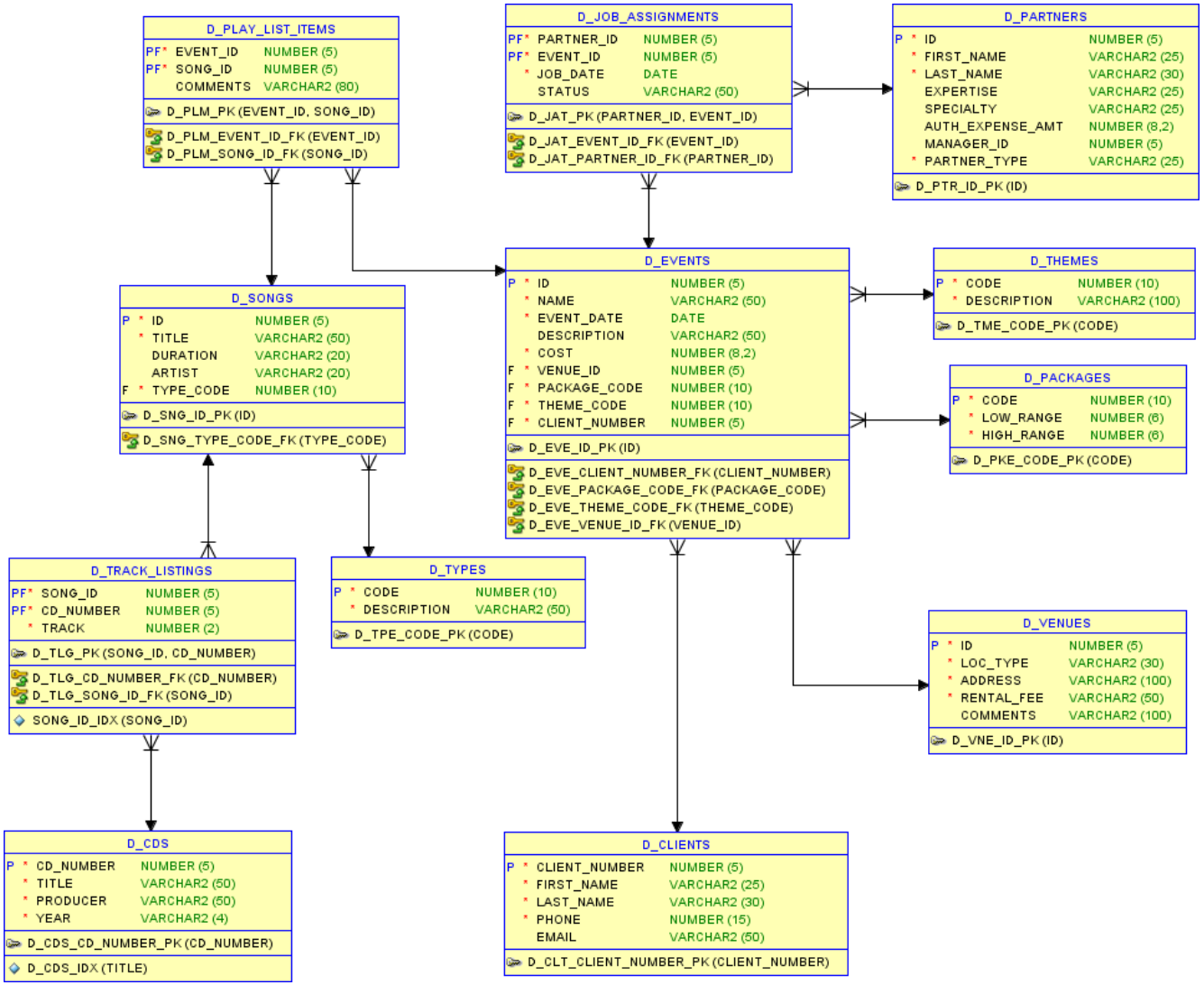
NANYANG THE INNOVATIVE POLYTECHNIC School of Information Technology

Database Schema for School Database



NANYANG THE INNOVATIVE POLYTECHNIC School of Information Technology

Database Schema for DJs on Demand



NANYANG THE INNOVATIVE POLYTECHNIC School of Information Technology

Database Schema for Global Fast Foods